ELSEVIER

Theoretical Computer Science 220 (1999) 31–65

Theoretical Computer Science

www.elsevier.com/locate/tcs

Using knowledge to optimally achieve coordination in distributed systems[1]

Gil Neiger[a], Rida A. Bazzi[b,*,2]

[a] *MicroComputer Research Labs, Intel Corporation, EY2-09, 5350 N. E. Elam Young Parkway, Hillsboro, OR 97124-6461, USA*

[b] *Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287-5406, USA*

Abstract

A distributed computing system consists of a set of individual processors that communicate through some medium. Coordinating the actions of such processors is essential in distributed computing. Researchers have long endeavored to find efficient solutions to a variety of coordination problems. Recently, processor *knowledge* has been used to characterize such solutions and to derive more efficient ones. Most of this work has concentrated on the relationship between *common knowledge* and *simultaneous* coordination. This paper considers non-simultaneous coordination problems. The results of this paper add to our understanding of the relationship between knowledge and the different requirements of coordination problems. This paper considers the ideas of *optimal* and *optimum* solutions to a coordination problem and precisely characterizes the problems for which optimum solutions exist. This characterization is based on combinations of *eventual common knowledge* and *continual common knowledge*. The paper then considers more general problems, for which optimal, but no optimum, solutions exist. It defines a new form of knowledge, called *extended knowledge*, which combines eventual and continual knowledge, and shows how extended knowledge can be used to both characterize and construct optimal protocols for coordination.

Keywords: Knowledge; Common knowledge; Distributed coordination; Optimal algorithms; Optimum algorithms

* Corresponding author. E-mail: bazzi@asu.edu.

[1] This work was supported in part by the National Science Foundation under grants CCR-8909663, CCR-9106627, and CCR-9301454. An earlier version of this paper appeared in: Yoram Moses (Ed.), *Proc. 4th Conference on Theoretical Aspects of Reasoning about knowledge*, Morgan-Kaufmann, Los Altos, CA, March 1992, pp. 43–59.

[2] This author was supported in part by a scholarship from the Hariri Foundation.

0304-3975/99/$ - see front matter
PII: S0304-3975(98)00236-9

1. Introduction

A distributed computing system consists of a set of individual processors that communicate through some medium. Coordinating the activity of such processors is fundamental in distributed computing. One way to achieve such coordination is to require the processors to agree on a common action to perform. In addition, processors must ensure that the action chosen is legitimate given the context within which they are operating (e.g., with respect to their initial values). The purpose of this work is to explore the relationship between knowledge and coordination and to use it to derive efficient solutions to coordination problems.

This work specifically considers *fault-tolerant* coordination in a distributed computing system. It is assumed that some (but not all) of the processors in the system may be faulty. A *coordination protocol* is an algorithm by which the nonfaulty processors successfully coordinate their actions despite the failures of others. There is a large body of literature that has studied fault-tolerant solutions to coordination problems, such as *Reliable Broadcast* and *Distributed Consensus* (Fischer [8] provides a survey of many such problems).

More recently, researchers have studied the relationship between *simultaneous* coordination and *common knowledge* [13]. Dwork and Moses [5] showed that achieving common knowledge was necessary for *Simultaneous Byzantine Agreement*. Moses and Tuttle [16] extended this result to a broad class of simultaneous coordination problems. Neiger and Tuttle [18] considered the class of *consistent* simultaneous coordination problems and showed that, in general, their solutions require a stronger form of common knowledge. The three papers above used the necessity of common knowledge to construct *optimum* coordination protocols.[3] By having processors perform actions as soon as the required knowledge is attained, these protocols are guaranteed to match or outperform any other solution. Michel [15] used the relationship between coordination and common knowledge to show that *Simultaneous Byzantine Agreement* cannot be achieved optimally in systems with arbitrary failures by polynomial-time coordination protocols.

The requirement of *simultaneous* coordination is very strong, and this is why common knowledge is needed to achieve it. But common knowledge is difficult to attain. Halpern and Moses [13] showed that it cannot be attained in many practical distributed systems and that, therefore, simultaneous coordination is impossible in these systems. In addition, the requirement of simultaneity is so strong as to obscure the relationship between knowledge and other requirements of coordination problems (see below). These facts motivate a study of the relationship between knowledge and problems requiring *nonsimultaneous* coordination. Halpern, Moses, and Waarts [14] considered one such problem, *Eventual Byzantine Agreement*, and developed a new form of knowledge, *continual common knowledge*, that could be used to develop *optimal* solutions.

[3] These papers referred to the protocols they developed as *optimal*. As will be seen below, there is an important distinction between optimum and optimal protocols.

Because the distinction between optimum and optimal solutions is central to many of the results of this paper, we describe it here briefly. Given two protocols P and P' that solve a coordination problem, P *dominates* P' if, in any environment, processors running P choose an action at least as early as those running P'. This gives a *partial* order on solutions: two solutions may be incomparable if each outperforms the other in some environment. A solution is *optimum* if it dominates all solutions; it is *optimal* if no solution strictly dominates it. In a given setting, a particular problem may or may not have an optimum solution, but it will always have one or more optimal solutions. It turns out that, for problems involving simultaneous coordination, there are often optimum solutions [5, 16, 18].

Previous work on knowledge and nonsimultaneous coordination considered protocols that guarantee that processor choices are *valid* (e.g., with respect to processors' initial states) and in *agreement* with each other. Their knowledge-based analyses did not explicitly consider a third requirement of most coordination problems, *termination*, even though termination is considered in coordination problems such as Eventual Byzantine Agreement. Such a requirement specifies the executions in which a nonfaulty processor *must* terminate the protocol by performing some action. Termination is thus a liveness property (as opposed to validity and agreement, which are safety properties). Ideally, a problem's termination condition would require all nonfaulty processors to perform an action in every execution. Unfortunately, this requirement cannot be achieved in many systems, including those that are asynchronous [10]. In this paper, we consider the weaker termination condition developed by Gopal and Toueg [11]. This condition requires that, in any execution in which some processor performs an action, all nonfaulty processors must do so also. To reason about this kind of termination, one needs to consider *eventual common knowledge* [13, 20].

This paper considers four types of coordination problems: complete and partial problems that require termination and those that do not. (Complete problems place requirements on the actions of all processors, while partial problems only constrain the correct processors.) For each, we establish the minimum knowledge necessary to perform an action. In some cases, this is simple knowledge or belief but, in the cases requiring termination, eventual common knowledge is required. We then consider the problem of deriving optimum solutions to these problems. Although some coordination problems have no optimum solution (e.g., *Eventual Byzantine Agreement*), such solutions do exist for others and this paper precisely characterizes those problems. This characterization uses both continual and eventual common knowledge; the type of knowledge used depends directly on the type of problem being considered.

We then consider *optimal* solutions, which do exist for all coordination problems. We show how these solutions can be characterized and constructed using different forms of knowledge. For problems requiring termination, this requires a new variant of common knowledge that combines the continual knowledge needed for agreement and the eventual knowledge needed for termination. We call this *extended knowledge*. The development and use of extended knowledge is one of the main contributions of this paper.

2. Definitions

This section defines a model of a distributed system. This model is similar to others used to study knowledge and coordination [5, 13, 14, 16, 18].

A distributed system is a finite set $\mathscr{P}$ of n processors and a communication network that connects them.

We consider both synchronous and asynchronous systems. In synchronous systems there are upper bounds on processors' speed and on message delivery delays. In asynchronous systems, no such bounds exist.

To describe a system execution, we assume the existence of an integer global clock. The global clock is not available to processors. At time 0, each processor starts in some *initial state*.[4] At each moment in time, zero, one, or more than one processors may take a step which consists of performing local computation (and, optionally, a coordination action), sending messages to other processors, and receiving messages delivered by the communication network. We do not model the receipt of messages by explicit events; rather, a message is available to a receiving processor the first time that processor is active after the message is delivered (a processor may fail to receive a delivered message; see below).

A processor's *local state* at any given time consists of its initial state, the sequence of steps it has taken, the messages it has sent and received, and the processor's identifier. A *global state* is a tuple $\langle s_1, \ldots, s_n \rangle$ of local states. A *run* of the system is an infinite sequence of global states, together with an operating environment (see below). An ordered pair $\langle r, t \rangle$, where r is a run and t is a natural number, is called a *point* and represents the state of the system at time t. The global state at point $\langle r, t \rangle$ is denoted by $r(t)$ and the local state of processor p at that point is denoted by $r_p(t)$. Processors follow a *communication protocol* P, which is a function of a processor's local state that specifies the messages a processor is to send (we assume that a processor sends no more than one message to another processor at any given time). One communication protocol may produce different runs depending on how the system behaves during execution. Important factors are the processors' initial states, how messages are delivered after they are sent, and how processors fail. Together these make up the *operating environment* of a run, described below.

Central to the results of this paper is the fact that one can compare the performance of different protocols with respect to the same behavior on the part of the "system". The *operating environment* of an execution captures this behavior. It includes all information (besides the protocol) necessary to reconstruct the execution. There are three components to an operating environment: the initial states of the processors, information about which processors are active at each instant of time, and information about message-transmission. Formally, an *operating environment* o is a triple $\langle \mathrm{I}, \mathrm{A}, \mathrm{M} \rangle$, where

[4] A processor's initial state is meant to model any input that a processor may receive from outside the system. Alternatively, inputs may be represented explicitly, allowing us to model inputs that may be received after time 0.

I is a vector of initial states (I[p] is the initial state of p), A is an activation pattern, and M is a message-transmission pattern, the latter two described below.

The *activation pattern* of a run specifies which processors are active at each instant of time. Formally, an activation pattern is a function $A : \mathbf{N} \mapsto 2^{\mathscr{P}}$. Processors in $A(t)$ are active at time t, while those in $\mathscr{P} - A(t)$ are not.

The *message-transmission pattern* of a run specifies the following information for every message that might be sent in a run: whether or not it is sent correctly; if and when it is delivered; and whether or not it is correctly received. Formally, a message-transmission pattern is a function $M : \mathbf{N} \times \mathscr{P} \times \mathscr{P} \mapsto \{Y, N\} \times \mathbf{N}^{\infty} \times \{Y, N\}$ (where $\mathbf{N}$ is the set of positive integers and $\mathbf{N}^{\infty} = \mathbf{N} \cup \{\infty\}$). The meaning of such a pattern is described in the following paragraphs.

Suppose that $M(t, p, q) = \langle b_s, t_d, b_r \rangle$. If $b_s = N$, then processor p does not send a message to q at time t. If p is active at that time and tries to send a message to q, it fails to send the message. The values of t_d and b_r are irrelevant in this case.

If $b_s = Y$ and p is active in time t and the protocol specifies for a message be sent at that time, then the message is sent correctly (if the protocol specifies for no message to be sent, then the value of $M(t, p, q)$ is defined but not does not affect the consistency of the operating environment with the run; see below).

If $b_s = Y$ and $t_d = \infty$, then the message (if sent) is lost by the communication network (a failure). The value of b_r is irrelevant in this case.

If $b_s = Y$ and $t_d \in \mathbf{N}$, then the message (if sent) is delivered at time $t_d > t$.[5] If $b_s = Y$, $t_d \in \mathbf{N}$, and $b_r = N$, then q omits to receive the message when it is delivered (a failure). If $b_s = Y$, $t_d \in \mathbf{N}$, and $b_r = Y$, then the message is correctly sent, delivered, and received. This means that q will have access to the message (i.e., the message will be reflected in q's local state) the next time q is active after t_d.

The message-transmission pattern allows failure by omission on the part of the sending processor, the communication network, or the receiving processor. Other failures (such as the duplication or corruption of a message) are not considered in this paper.

The classes of allowable activation and message-transmission patterns are determined by assumptions made about the communication network and about processor failures. For example, message-passing may be *synchronous* (there is an upper bound on $t_d - t$), or *asynchronous* (there is no upper bound on $t_d - t$). It may be reliable (all messages are delivered) or lossy (messages may be lost). Any of these restrictions can be specified by restricting the class of activation and message-transmission patterns that can occur.

Different processor failure models place different restrictions on the faulty behaviors that processors may exhibit through the activation and message-transmission patterns (a formal specification of the failure patterns allowed by different failure models is beyond the scope of this paper). We can model any type of *benign* failures, specifically crash (stopping) failures, send-omission failures, and general omission failures [17]. We cannot, however, model arbitrary processor failures [19]. Given a run r with operating

[5] We do not allow $t_d \leqslant t$.

environment $o = \langle \text{I}, \text{M}, \text{A} \rangle$, one can characterize the set $\mathcal{N}(r)$ of processors that are not faulty in r:

$$\begin{aligned}\mathcal{N}(r) = \{\ p \in \mathcal{P} \mid & (\forall t \in \mathbf{N} \mid (\exists t' \in \mathbf{N} \mid t' > t : p \in \text{A}(t')) \wedge \\ & (\forall q_{\text{r}} \in \mathcal{P} \mid (\exists t_d \in \mathbf{N}^{\infty} \mid (\exists b_r \in \{\text{Y}, \text{N}\} \mid \text{M}(t, p, q_{\text{r}}) = \langle \text{Y}, t_d, b_r \rangle))) \wedge \\ & (\forall q_{\text{s}} \in \mathcal{P} \mid (\exists b_s \in \{\text{Y}, \text{N}\} \mid (\exists t_d \in \mathbf{N}^{\infty} \mid \text{M}(t, q_{\text{s}}, p) = \langle b_s, t_d, \text{Y} \rangle)))))\}.\end{aligned}$$

The first conjunct specifies that a non-faulty processor is active infinitely often. The second specifies that the processor correctly sends every message it should. The third specifies that it receives every message delivered to it. For the remainder of the paper, we assume that for no run r does $\mathcal{N}(r) = \emptyset$.

The operating environment of a run must be consistent with the run in that it must match the initial states and message deliveries of the run. For example, consider an operating environment o that indicates that p does not send a message to q at time t (i.e., that $\text{M}(t, p, q) = (\text{N}, t_d, b_r)$ for some t_d and b_r). Suppose that, in run r, p sends such a message at time t; operating environment o is *not* consistent with r. On the other hand, if p did not send such a message (either because it omitted to do or because its protocol did not call for such a message to be sent), then o (as described) is consistent.[6] We assume that the operating environment of every run is consistent with the run.

Two runs of two different communication protocols *correspond* (or are *corresponding runs*) if they have the same operating environment. Different protocols are compared by comparing their behavior in corresponding runs. Note that a run of one protocol may correspond to more than one run of another protocol. For example, suppose that in a run of P_1, processor p is active and sends a message to q at time t, and that in a run of P_2, p is active, but sends no message to q. Consider two runs of P_1: in one, p correctly sends the message and, in the other, p omits to send it. Both of these runs may correspond to the same run of P_2; this is because the operating environment's specification regarding this message is not relevant to P_2.

This work identifies a system with the set of all runs of a communication protocol under a given failure model and with specified assumptions about message-passing. Such a set of runs is denoted by $\mathcal{R}_P$, where P is the communication protocol being used; $\mathcal{R}$ will be used if P is obvious from context. If $r \in \mathcal{R}$ and t is a natural number, then $\langle r, t \rangle$ is a *point in* $\mathcal{R}$. In order to analyze systems, we define a logical language in which one can make statements about the system. A *fact* in this language is interpreted to be a property of points: a fact φ will be either true or false at a given point $\langle r, t \rangle$ in $\mathcal{R}$, denoted $(\mathcal{R}, r, t) \models \varphi$ and $(\mathcal{R}, r, t) \not\models \varphi$, respectively. Fact φ is *valid in system* $\mathcal{R}$, denoted $\mathcal{R} \models \varphi$, if it is true at all points in $\mathcal{R}$. A fact is *valid* if it is valid in all systems. Although facts are interpreted as properties of points, it is often convenient

[6] One could give a formal definition of the consistency of a run with an operating environment. However, doing so would require more complete formal definitions of, for example, a communication protocol. Doing so would be straightforward but would add a significant amount of technical detail that is not directly relevant to the main results of the paper.

to refer to facts that are about objects other than points (e.g., properties of runs). In general, a fact φ is a *fact about* X if fixing X determines the truth (or falsity) of φ. A fact φ is *stable in* $\mathscr{R}$ if, once it becomes true it remains so; for all points $\langle r,t\rangle$ in $\mathscr{R}$, if $(\mathscr{R},r,t)\models\varphi$, then $(\mathscr{R},r,t')\models\varphi$ for all $t'\geq t$.

3. Coordination problems

This section defines four classes of coordination problems using the model given in the previous sections. Informally, a coordination problem requires processors to coordinate by choosing a common action from a set of possible actions. In any given context, some subset of the possible actions are *enabled*, and processors should only choose an enabled action. Formally, a *coordination problem* is a finite set of actions $\mathscr{C}=\{a_1,\ldots,a_m\}$ together with a set of associated *enabling conditions* $\{ok_1,\ldots,ok_m\}$. Each enabling condition is a fact about the initial input and the identities of the faulty processors (thus, it is a fact about runs) so that processors actions can depend on what initial values other processors have and any knowledge of failures that processors might have. The processors must coordinate to choose a common action that is enabled. Processors need not perform their actions simultaneously. Formal specifications of the correctness of a protocol to achieve coordination are given below. Given an enabling condition, it is not always the case that a processor can tell whether it holds or not. For example, in an asynchronous system, it is not possible to tell at any given time if a processor has crashed. Nevertheless, it is sometimes possible to detect message loss in an asynchronous system if messages are guaranteed to be delivered in the order they are sent.

An example of a coordination problem is *Eventual Byzantine Agreement*, which was considered by Halpern, Moses, and Waarts [14]. In this problem, all processors begin with an initial value that is either 0 or 1. Each processor must eventually decide on a final value, also 0 or 1. All processors must choose the same final value and this value must be the initial value of some processor. This can be cast in our framework by having two actions: a_0 corresponds to selecting 0 as a final value and a_1 corresponds to selecting 1. The enabling conditions are given as follows: ok_0 holds if some processor had initial value 0 and ok_1 holds if some processor had initial value 1.

For a protocol to coordinate a choice of actions, there must be a mechanism by which it can specify when an action is to be performed. An *action protocol* $P(\Phi)$ is a communication protocol P augmented by an *action function* Φ. For each $a_i\in\mathscr{C}$ and $p\in\mathscr{P}$, $\Phi_{i,p}$ is a fact about p's local state (see Section 2 above). $P(\Phi)$ has p perform a_i the first time at which $\Phi_{i,p}$ is true and p is active (note that, since $\Phi_{i,p}$ is a stable fact about p's local state, p can always tell when the fact first becomes true for it). An action protocol $P(\Phi)$ is a *decision protocol* if the following two properties hold of the action function Φ for all $\mathscr{R}_P$:

- For all $a_i\in\mathscr{C}$ and $p\in\mathscr{P}$, $\Phi_{i,p}$ is stable in $\mathscr{R}_P$; that is, a processor's choice is irrevocable.

- For all $a_i \in \mathscr{C}$ and $p \in \mathscr{P}$, if $(\mathscr{R}_P, r, t) \models \Phi_{i,p}$, then for no $j \neq i$ and $t' \in \mathbf{N}$ does $(\mathscr{R}_P, r, t') \models \Phi_{j,p}$; that is, a processor's choice is unique.

Note that the action function Φ is completely orthogonal to the communication protocol P. Although it is triggered by a processor's local state, it does not modify that state (a processor can tell that it executed an action because the enabling fact is stable). It controls only the actions of a processor, which are, technically speaking, not part of its local state. A processor does not stop running its communication protocol once it has made a choice. For this reason, the system $\mathscr{R}_P$ is well-defined as the set of runs of the communication protocol P and is independent of any action function.

There are various ways to define the correctness of an action protocol with respect to a problem $\mathscr{C}$. Informally, processors must agree and must choose an action that is enabled. In some cases, the actions taken by the *faulty* processors are not relevant; in others, their actions are subject to the same correctness criteria as those of the nonfaulty processors. We call these cases *partial* and *complete*, respectively, and discuss them in the next two sections. The earlier literature on knowledge and coordination concentrated on partial coordination [14, 16]; more recently, researchers have begun to consider complete coordination [11, 18].[7]

3.1. Partial coordination

Most coordination problems defined in the literature require only that the nonfaulty processors coordinate their actions. For example, *Eventual Byzantine Agreement* places no restrictions on the values that may be chosen by the faulty processors [4]. Such problems are easier to solve than problems that place restrictions on the actions of faulty processors. In systems in which processors can fail by taking arbitrary actions, only partial coordination problems can be solved because there is no way to force faulty processors to choose an enabled action.

Formally, $P(\Phi)$ *partially satisfies* $\mathscr{C}$ (or *P-satisfies* $\mathscr{C}$) if the following hold:

- Validity. If an action is performed by a nonfaulty processor, then that action is enabled: $\mathscr{R}_P \models \Phi_{i,p} \wedge (p \in \mathscr{N}) \Rightarrow ok_i$.
- Agreement. If two nonfaulty processors perform actions, they perform the same action: if $(\mathscr{R}_P, r, t_p) \models \Phi_{i,p} \wedge (p \in \mathscr{N})$ and $(\mathscr{R}_P, r, t_q) \models \Phi_{i,q} \wedge (q \in \mathscr{N})$, then $i = j$.

Note that one can trivially solve any coordination problem as specified above simply by choosing $\Phi_{i,p} = \mathit{false}$ for all $a_i \in \mathscr{C}$ and $p \in \mathscr{P}$. This is because Validity and Agreement are safety properties. Nevertheless, it still makes sense to consider optimum and optimal solutions to such problems (see below). In most cases, such solutions would not be trivial. Coordination problems can also be made nontrivial by adding a liveness property such as a *termination condition*.

[7] Neiger and Tuttle [18] referred to such problems as *consistent* coordination problems.

Informally, a termination condition specifies when nonfaulty processors *must* perform an action (because of failures, one cannot require faulty processors to perform an action). Some problems require that all nonfaulty processors perform an action in every run. However, these problems cannot, in general, be solved in systems with asynchronous communication [10]. For that reason, Gopal and Toueg [11] introduced a weaker termination condition, which requires nonfaulty processors to act only if some other processor does. Formally, $P(\Phi)$ *partially satisfies* $\mathscr{C}$ *with termination* (or *PT-satisfies* $\mathscr{C}$) if it P-satisfies $\mathscr{C}$ and the following condition holds:

- Termination. If a nonfaulty processor performs an action, then all nonfaulty processors perform that action: if $(\mathscr{R}_P, r, t) \models \Phi_{i,p} \wedge (p \in \mathscr{N})$, then, for each $q \in \mathscr{N}(r)$, there is some t' such that $(\mathscr{R}_P, r, t') \models \Phi_{i,q}$.

If $P(\Phi)$ X-satisfies $\mathscr{C}$ (where X is either P or PT), then we say that $P(\Phi)$ is an *X-solution to* $\mathscr{C}$.

Halpern, Moses, and Waarts compared partial solutions to *Eventual Byzantine Agreement* by comparing their behavior in corresponding runs (i.e., in runs with the same operating environment). Their method is adapted here. Suppose that decision protocols $P_1(\Phi_1)$ and $P_2(\Phi_2)$ both P-satisfy some problem $\mathscr{C}$. $P_1(\Phi_1)$ *P-dominates* $P_2(\Phi_2)$ if, in every pair (r_1, r_2) of corresponding runs of the two protocols, $P_2(\Phi_2)$ has no nonfaulty processor perform an action earlier in r_2 than $P_1(\Phi_1)$ does in r_1 (it may be that some processors perform actions in neither run). Formally, if $P_1(\Phi_1)$ P-dominates $P_2(\Phi_2)$ and r_1 and r_2 are corresponding runs of the two protocols, then $(\mathscr{R}_{P_2}, r_2, t) \models \Phi_{2(i,p)}$ implies $(\mathscr{R}_{P_1}, r_1, t) \models \bigvee_{a_j \in \mathscr{C}} \Phi_{1(j,p)}$ for all $a_i \in \mathscr{C}$ and $p \in \mathscr{N}(r_1)$ (since r_1 and r_2 are corresponding, $\mathscr{N}(r_1) = \mathscr{N}(r_2)$).

Notice that the P-dominates relation is a partial order on the space of P-solutions to a given problem. It may be that neither of $P_1(\Phi_1)$ and $P_2(\Phi_2)$ P-dominates the other; $P_1(\Phi_1)$ may outperform $P_2(\Phi_2)$ in one operating environment, while $P_2(\Phi_2)$ outperforms $P_1(\Phi_1)$ in another.

A protocol is *P-optimum* (respectively, *PT*-optimum) *for* $\mathscr{C}$ if it P-satisfies (respectively, *PT*-satisfies) $\mathscr{C}$ and P-dominates every other protocol that does so. Because the P-dominates order is partial, some problems may not have optimum solutions. For example, Moses and Tuttle [16] gave two incomparable P-solutions to *Eventual Byzantine Agreement*: in one solution processors can decide on 0 very quickly, but are slow in deciding 1; in the other solution, processors can decide 1 very quickly, but are slow in deciding 0. They showed that there is no P-solution such that processors can decide both values very quickly. Thus, there is no P-solution that P-dominates both of the solutions described above and, hence, there is no P-optimum protocol. In contrast, protocol $P(\Phi)$ is *P-optimal for* $\mathscr{C}$ (respectively, *PT*-optimal) if it P-satisfies $\mathscr{C}$ (respectively, *PT*-satisfies) and if every P-solution to $\mathscr{C}$ (respectively, *PT*-solution) that P-dominates $P(\Phi)$ is in turn P-dominated by $P(\Phi)$.

Although there is no optimum solution to *Eventual Byzantine Agreement*, there are coordination problems for which optimum solutions do exist. Section 6 precisely characterizes these problems. Section 9 shows how to construct optimal solutions to any of the problems defined here.

3.2. Complete coordination

Although partial coordination is appropriate for systems in which processors can fail arbitrarily, much literature has studied systems with relatively benign failures. In these systems, it is appropriate to study coordination problems in which the actions of faulty processors (if any) must be consistent with those of the correct processors. This section defines a class of complete coordination problems that put additional restrictions on the class of partial coordination problems of the previous section.

$P(\Phi)$ *completely satisfies* $\mathscr{C}$ (or *C-satisfies* $\mathscr{C}$) if the following hold:

- Validity. If an action is performed by any processor, then that action is enabled: $\mathscr{R}_P \models \Phi_{i,p} \Rightarrow ok_i$.
- Agreement. If two processors perform actions, they perform the same action: if $(\mathscr{R}_P, r, t_p) \models \Phi_{i,p}$ and $(\mathscr{R}_P, r, t_q) \models \Phi_{j,q}$, then $i = j$.

$P(\Phi)$ *completely satisfies* $\mathscr{C}$ *with termination* (or *CT*-satisfies $\mathscr{C}$) if it *C*-satisfies $\mathscr{C}$ and the following condition holds:

- Termination. If any processor performs an action, then all nonfaulty processors perform that action: if $(\mathscr{R}_P, r, t) \models \Phi_{i,p}$, then, for each $q \in \mathscr{N}(r)$, there is some t' such that $(\mathscr{R}_P, r, t') \models \Phi_{i,q}$.

If $P(\Phi)$ *X*-satisfies $\mathscr{C}$ (where *X* is either *C* or *CT*), then we say that $P(\Phi)$ is an *X*-solution to $\mathscr{C}$.

If $P_1(\Phi_1)$ and $P_2(\Phi_2)$ are *C*-solutions to $\mathscr{C}$, then $P_1(\Phi_1)$ *C-dominates* $P_2(\Phi_2)$ if, for all pairs r_1, r_2 of corresponding runs of the two protocols, $(\mathscr{R}_{P_2}, r_2, t) \models \Phi_{i,p}$ implies $(\mathscr{R}_{P_1}, r_1, t) \models \bigvee_{a_j \in \mathscr{C}} \Phi_{j,p}$ for all $a_i \in \mathscr{C}$ and $p \in \mathscr{P}$ (notice that this must be true for all p, including those that are faulty).

A protocol is *C-optimum for* $\mathscr{C}$ if it *C*-satisfies $\mathscr{C}$ and *C*-dominates every other protocol that does so. $P(\Phi)$ is *C-optimal for* $\mathscr{C}$ if it *C*-satisfies $\mathscr{C}$ and if every *C*-solution to $\mathscr{C}$ that *C*-dominates $P(\Phi)$ is in turn *C*-dominated by $P(\Phi)$.

Similarly, a protocol is *CT-optimum for* $\mathscr{C}$ if it *CT*-satisfies $\mathscr{C}$ and *C*-dominates every other protocol that does so. $P(\Phi)$ is *CT-optimal for* $\mathscr{C}$ if it *CT*-satisfies $\mathscr{C}$ and if every *CT*-solution to $\mathscr{C}$ that *C*-dominates $P(\Phi)$ is in turn *C*-dominated by $P(\Phi)$.

4. Definitions of knowledge

The analysis in this paper depends on a processor's knowledge at different points in an execution. This section formally defines processor knowledge. The treatment here is an adaptation of others [5, 13, 14, 16, 18].

4.1. Basic definitions

This section gives a way to express processor knowledge by augmenting the logical language introduced in Section 2. Recall that a *fact* in this language is a property of points: a fact φ is either true or false at a given point $\langle r, t \rangle$ in system $\mathscr{R}$, denoted $(\mathscr{R}, r, t) \models \varphi$ or $(\mathscr{R}, r, t) \not\models \varphi$, respectively. We assume that the language is powerful

enough to represent all relevant *ground* facts – facts about the system that do not explicitly mention processors' knowledge – and is closed under the standard boolean connectives.

Processor knowledge was first defined by Halpern and Moses [13] in the following way. *Processor p knows φ at point $\langle r,t\rangle$ in system $\mathscr{R}$*, denoted $(\mathscr{R},r,t)\models \mathsf{K}_p\varphi$, if $(\mathscr{R},r',t')\models\varphi$ for all points $\langle r',t'\rangle$ in $\mathscr{R}$ such that $r'_p(t')=r_p(t)$. Thus, a processor always knows any true fact about its local state (recall that an action protocol's predicates $\Phi_{i,p}$ are all facts about p's local state).

Because this paper deals with coordination among a group of processors, different forms of *group knowledge* are important. The particular setting (e.g., type of coordination problem) determines the group whose knowledge is of interest. We often consider sets of processors whose membership may vary from one run to another or over the course of a run. These are called *indexical sets*; their membership is determined by the point being considered. For example, if $\mathscr{S}$ is an indexical set, then $\mathscr{S}(r,t)$ refers to the contents of the set at point $\langle r,t\rangle$. Examples of indexical sets include the set of processors that know a certain fact or that have performed a certain action. On occasion, we will abuse notation slightly and use $\mathscr{N}$ to represent the *indexical* set of nonfaulty processors. It should be understood in these cases that $\mathscr{N}(r,t)$ is equal to $\mathscr{N}(r)$, the set of processors nonfaulty throughout run r, and *not* the (potentially larger) set of processors that have behaved correctly through time t of run r.

It is often useful to condition a processor's knowledge on the processor's membership in a specific set. We say that *processor p believes φ conditional on $\mathscr{S}$* if p knows that, if it is in $\mathscr{S}$, φ is true. That is,

$$\mathsf{B}_p^{\mathscr{S}}\varphi \equiv \mathsf{K}_p(p\in\mathscr{S}\Rightarrow\varphi).$$

It is easy to see that $(\mathscr{R},r,t)\models \mathsf{B}_p^{\mathscr{S}}\varphi$ if $(\mathscr{R},r',t')\models\varphi$ for all points $\langle r',t'\rangle$ such that $r'_p(t')=r_p(t)$ and $p\in\mathscr{S}(r',t')$. Processor knowledge, using the K_p operators, will be used to define strong notions of group knowledge, while processor belief, using $\mathsf{B}_p^{\mathscr{S}}$, will be used to define weaker notions.

Informally, a fact φ is *common knowledge* to $\mathscr{S}$ if everyone in $\mathscr{S}$ knows φ, everyone knows that everyone knows φ, and so on.[8] Common knowledge is necessary for the solution to simultaneous coordination problems [5, 16, 18]. The following is a brief overview of a more formal definition, based on logical fixed points. *Everyone in indexical set $\mathscr{S}$ knows φ*, denoted $\mathsf{E}_{\mathscr{S}}\varphi$, is defined to be $\bigwedge_{p\in\mathscr{S}}\mathsf{K}_p\varphi$. *All processors in $\mathscr{S}$ believe φ*, denoted $\mathsf{A}_{\mathscr{S}}\varphi$, is equivalent to $\bigwedge_{p\in\mathscr{S}}\mathsf{B}_p^{\mathscr{S}}\varphi$. Based on this, two forms of common knowledge are defined, a strong one based on knowledge and a weak one based on belief. *Strong common knowledge of fact φ by set $\mathscr{S}$*, denoted $\mathsf{S}_{\mathscr{S}}\varphi$, is the greatest fixed point of the equation

$$X\equiv \mathsf{E}_{\mathscr{S}}(\varphi\wedge X).$$

[8] When necessary to distinguish it from other forms of knowledge, we may refer to this as simple common knowledge.

Fagin et al. [6] provide an extensive discussion of fixed points and their relevance to the study of knowledge. The following is a brief summary.

Saying that $\mathsf{S}_{\mathcal{S}}\varphi$ is the greatest fixed point of $X \equiv \mathsf{E}_{\mathcal{S}}(\varphi \wedge X)$ means two things:

- $\mathsf{S}_{\mathcal{S}}\varphi$ is a solution to the equation, that is, $\mathsf{S}_{\mathcal{S}}\varphi \equiv \mathsf{E}_{\mathcal{S}}(\varphi \wedge \mathsf{S}_{\mathcal{S}}\varphi)$, and
- $\mathsf{S}_{\mathcal{S}}\varphi$ is *greater* than all other fixed points in the sense that, if ψ is also a fixed point (i.e., $\psi \equiv \mathsf{E}_{\mathcal{S}}(\varphi \wedge \psi)$), then $\psi \Rightarrow \mathsf{S}_{\mathcal{S}}\varphi$.

Fagin et al. show that the equation $X \equiv F(X)$ has a greatest fixed point if $F(X)$ is *monotone* [6, Lemma 11.5.4]. $F(X)$ is monotone if the validity of $\psi \Rightarrow \theta$ implies that $F(\psi) \Rightarrow F(\theta)$ is also valid. The following is a proof that will serve as a model for all future statements of monotonicity in this paper:

Theorem 1. *The function* $\mathsf{E}_{\mathcal{S}}(\varphi \wedge X)$ *is monotone.*

Proof. Assume that $\psi \Rightarrow \theta$ is valid, that is, that θ is true at any point at which ψ is true. The proof must show that $\mathsf{E}_{\mathcal{S}}(\varphi \wedge \psi) \Rightarrow \mathsf{E}_{\mathcal{S}}(\varphi \wedge \theta)$ is also valid. Assume that $\mathsf{E}_{\mathcal{S}}(\varphi \wedge \psi)$ holds at point $\langle r, t\rangle$. The proof must show that $\mathsf{E}_{\mathcal{S}}(\varphi \wedge \theta)$ also holds at $\langle r, t\rangle$. This means that, for any $p \in \mathcal{S}(r,t)$ and $\langle r', t'\rangle$ with $r'_p(t') = r_p(t)$, $\varphi \wedge \theta$ is true at $\langle r', t'\rangle$. Since $\mathsf{E}_{\mathcal{S}}(\varphi \wedge \psi)$ holds at $\langle r, t\rangle$, $\varphi \wedge \psi$ holds at $\langle r', t'\rangle$. Since $\psi \Rightarrow \theta$ is valid, $\varphi \wedge \theta$ holds at $\langle r', t'\rangle$. □

Corollary 2. $\mathsf{S}_{\mathcal{S}}\varphi$ *is well defined.*

Weak common knowledge of fact φ *by set* $\mathcal{S}$, denoted $\mathsf{W}_{\mathcal{S}}\varphi$, is the greatest fixed point of the equation

$$X \equiv \mathsf{A}_{\mathcal{S}}(\varphi \wedge X).$$

Note that the function $\mathsf{A}_{\mathcal{S}}(\varphi \wedge X)$ is also monotone, so $\mathsf{W}_{\mathcal{S}}\varphi$ is well defined. Using techniques of Fagin et al., it is easy to show that $\mathsf{S}_{\mathcal{S}}\varphi$ is equivalent to the infinite conjunction $\bigwedge_{i \geqslant 1} \mathsf{E}^i_{\mathcal{S}}\varphi$ and that $\mathsf{W}_{\mathcal{S}}\varphi$ is equivalent to $\bigwedge_{i \geqslant 1} \mathsf{A}^i_{\mathcal{S}}\varphi$.

Neiger and Tuttle [18] showed that for complete simultaneous coordination, processors must have strong common knowledge that an enabling condition is true (they called this *consistent* simultaneous coordination). In fact, for simultaneous coordination, if a processor acts, it must know that every other processor is acting at the same time. This means that each processor must know that every other processor knows that it is OK to act, and so on. Moses and Tuttle [16] had earlier observed that achieving weak common knowledge was sufficient to achieve partial simultaneous coordination (they did not consider complete coordination).

The remainder of this section introduces two modifications of common knowledge that are appropriate to the study of nonsimultaneous coordination. Each has a strong and a weak version, which are appropriate to the analysis of complete and partial coordination problems, respectively.

4.2. Eventual common knowledge

Eventual common knowledge [13, 20] relaxes the simultaneity that is inherent in simple common knowledge. For this reason, it is appropriate in the study of problems that do not require simultaneous coordination. Informally, a fact is eventual common knowledge to a set of processors if they all eventually know it, all eventually know that all others eventually know it, and so forth. As will be seen below, eventual common knowledge is necessary for achieving termination in solutions to coordination problems.

The definition of eventual knowledge uses the temporal operator *eventually* $\Diamond$. $(\mathscr{R}, r, t) \models \Diamond\varphi$ if and only if $(\mathscr{R}, r, t') \models \varphi$ for some $t' \geq t$.[9] Eventual common knowledge is defined in a manner analogous to that of simple common knowledge. *Strong eventual common knowledge of fact* φ *by set* $\mathscr{S}$, denoted $\mathsf{S}^{\Diamond}_{\mathscr{S}}\varphi$, is the greatest fixed point of the equation

$$X \equiv \Diamond \mathsf{E}_{\mathscr{S}}(\varphi \wedge X).^{10}$$

Weak eventual common knowledge of fact φ *by set* $\mathscr{S}$, denoted $\mathsf{W}^{\Diamond}_{\mathscr{S}}\varphi$, is the greatest fixed point of the equation

$$X \equiv \Diamond \mathsf{A}_{\mathscr{S}}(\varphi \wedge X).$$

Note that both $\Diamond\mathsf{E}_{\mathscr{S}}(\varphi \wedge X)$ and $\Diamond\mathsf{A}_{\mathscr{S}}(\varphi \wedge X)$ are monotone, so both forms of eventual common knowledge are well defined.

It is easy to see that $\mathsf{S}^{\Diamond}_{\mathscr{S}}\varphi$ implies the infinite conjunction $\bigwedge_{i \geq 1}(\Diamond\mathsf{E}_{\mathscr{S}})^i\varphi$. Similarly, $\mathsf{W}^{\Diamond}_{\mathscr{S}}\varphi$ implies $\bigwedge_{i \geq q}(\Diamond\mathsf{A}_{\mathscr{S}})^i\varphi$. One should note that eventual common knowledge is *weaker* than simple common knowledge. It does not require that processors gain their knowledge simultaneously or that more than one level of knowledge will ever hold simultaneously. Eventual common knowledge does not, in general, imply "eventually" common knowledge.

Both forms of eventual common knowledge satisfy *positive introspection*; if a fact is eventual common knowledge to a set, then all members of the set eventually know (or believe) this. Thus, the following are valid:

- $\mathsf{S}^{\Diamond}_{\mathscr{S}}\varphi \Rightarrow \Diamond\mathsf{E}_{\mathscr{S}}\mathsf{S}^{\Diamond}_{\mathscr{S}}\varphi$; and
- $\mathsf{W}^{\Diamond}_{\mathscr{S}}\varphi \Rightarrow \Diamond\mathsf{A}_{\mathscr{S}}\mathsf{W}^{\Diamond}_{\mathscr{S}}\varphi$.

These implications follow directly from the fixed-point definitions (In general the implications in the other direction do not hold). Each form of eventual common knowledge satisfies an induction rule that can be used to show that certain facts are eventual common knowledge:

- If $\varphi \Rightarrow \Diamond\mathsf{E}_{\mathscr{S}}(\varphi \wedge \psi)$ is valid in a system, then $\varphi \Rightarrow \mathsf{S}^{\Diamond}_{\mathscr{S}}\psi$ is also valid in that system.

[9] Note that these are *linear-time* semantics for $\Diamond$.

[10] This is slightly different from the original definition of Halpern and Moses [13]. They defined eventual common knowledge to be the greatest fixed point of the equation $X \equiv \bigwedge_{p \in \mathscr{S}} \Diamond\mathsf{K}_p(\varphi \wedge X)$. For all cases considered in this paper, their definition is equivalent to the one given here for strong eventual common knowledge. The definition given here is simpler and is consistent with the definitions of the other forms of common knowledge given in this paper.

• If $\varphi \Rightarrow \Diamond \mathsf{A}_{\mathcal{S}}(\varphi \wedge \psi)$ is valid in a system, then $\varphi \Rightarrow \mathsf{W}^{\Diamond}_{\mathcal{S}}\psi$ is also valid in that system. (Again, these follow from the fixed-point definitions.) This paper considers cases in which facts φ about runs (specifically, the enabling conditions of a coordination problem) become eventual common knowledge to the set $\mathcal{N}$ of nonfaulty processors. Because we consider only systems in which this set is never empty, it is not hard to see that, in these systems, $\mathsf{S}^{\Diamond}_{\mathcal{N}}\varphi \Rightarrow \varphi$ and $\mathsf{W}^{\Diamond}_{\mathcal{N}}\varphi \Rightarrow \varphi$ are valid if φ is a fact about the run. These implications will simplify the presentation of some protocols below.

4.3. Continual common knowledge

Although eventual common knowledge is necessary for Termination (as shown in Section 5 below), Halpern et al. [14] showed that it is not sufficient to enforce Agreement. Intuitively, the reason for this is that, unlike simple common knowledge, different processors may learn of eventual common knowledge at different times. If the enabling conditions of two actions both become eventual common knowledge, two processors may learn of them in different orders and, perhaps, perform different actions. This would violate Agreement. Halpern, Moses, and Waarts showed that, to ensure that no disagreement occurred at any time during a run, it is necessary to use a kind of knowledge that was continual over all points of a run. They called this *continual common knowledge*.

This form of knowledge makes use of the temporal operator *always* $\boxdot$. This is a bidirectional variant of the temporal operator *henceforth* $\Box$. $(\mathcal{R}, r, t) \models \boxdot\varphi$ if and only if $(\mathcal{R}, r, t') \models \varphi$ for all t'. Continual common knowledge can now be defined in a familiar manner. *Strong continual common knowledge of fact φ by set $\mathcal{S}$*, denoted $\mathsf{S}^{\boxdot}_{\mathcal{S}}\varphi$, is the greatest fixed point of

$$X \equiv \boxdot \mathsf{E}_{\mathcal{S}}(\varphi \wedge X).$$

Weak continual common knowledge of fact φ by set $\mathcal{S}$, denoted $\mathsf{W}^{\boxdot}_{\mathcal{S}}\varphi$ is the greatest fixed point of

$$X \equiv \boxdot \mathsf{A}_{\mathcal{S}}(\varphi \wedge X).$$

(As in the above cases, it is easy to see that both $\boxdot \mathsf{E}_{\mathcal{S}}(\varphi \wedge X)$ and $\boxdot \mathsf{A}_{\mathcal{S}}(\varphi \wedge X)$ are monotone, so the two forms of continual common knowledge are well defined.)

Halpern, Moses, and Waarts note that $\mathsf{W}^{\boxdot}_{\mathcal{S}}\varphi$ is equivalent to $\bigwedge_{i \geq 1}(\boxdot \mathsf{A}_{\mathcal{S}})^i\varphi$; similarly, $\mathsf{S}^{\boxdot}_{\mathcal{S}}\varphi$ is equivalent to the infinite conjunction $\bigwedge_{i \geq 1}(\boxdot \mathsf{E}_{\mathcal{S}})^i\varphi$. That is, a fact φ is continual common knowledge to a set if it is always the case that everyone in the set knows φ, it is always the case that everyone in the set knows that it is always the case that everyone in the set knows φ, etc. Continual common knowledge is *stronger* than simple common knowledge. It guarantees that all members know a fact at all times and that all levels of knowledge hold at all times. Continual common knowledge implies "continually" common knowledge. Halpern, Moses, and Waarts used weak continual common knowledge to construct optimal partial solutions to *Eventual Byzantine*

Agreement. Among other things, the current paper explores the use of strong continual common knowledge in the complete solutions of coordination problems.

It may seem odd that a stronger form of knowledge (continual common knowledge) is necessary to solve a weaker problem (nonsimultaneous coordination). One can explain this apparent contradiction by considering the sets of processors whose knowledge is relevant. To achieve simultaneous coordination, the entire set of nonfaulty processors must have common knowledge of some enabling condition. For nonsimultaneous coordination, only a subset (defined in Sections 6 and 8 below) of the nonfaulty processors must have continual common knowledge of an enabling condition. Because the set of processors involved is smaller, the required knowledge is easier to achieve.

The two forms of continual common knowledge have properties similar to eventual common knowledge. Both satisfy positive introspection; if a fact is continual common knowledge to a set, then it is always the case that all members of the set know this. Thus, the following are valid:

- $\mathsf{S}^{\boxdot}_{\mathscr{S}}\varphi \Rightarrow \boxdot \mathsf{E}_{\mathscr{S}}(\varphi \wedge \mathsf{S}^{\boxdot}_{\mathscr{S}}\varphi)$;
- $\mathsf{W}^{\boxdot}_{\mathscr{S}}\varphi \Rightarrow \boxdot \mathsf{A}_{\mathscr{S}}(\varphi \wedge \mathsf{W}^{\boxdot}_{\mathscr{S}}\varphi)$.

(These follow directly from the fixed-point definition.) Both satisfy a kind of negative introspection:

Theorem 3. *The following are valid*:

- $\neg \mathsf{S}^{\boxdot}_{\mathscr{S}}\varphi \wedge \mathsf{K}_p(p \in \mathscr{S}) \Rightarrow \mathsf{K}_p \neg \mathsf{S}^{\boxdot}_{\mathscr{S}}\varphi$;
- $\neg \mathsf{W}^{\boxdot}_{\mathscr{S}}\varphi \wedge (p \in \mathscr{S}) \Rightarrow \mathsf{B}^{\mathscr{S}}_p \neg \mathsf{W}^{\boxdot}_{\mathscr{S}}\varphi$.

Proof. The proofs of the two parts are similar, and only the first is presented here. Consider $p \in \mathscr{P}$ and some point $\langle r,t\rangle$ such that $(\mathscr{R},r,t) \models \neg \mathsf{S}^{\boxdot}_{\mathscr{S}}\varphi \wedge \mathsf{K}_p(p \in \mathscr{S})$. It suffices to show that, for any point $\langle r',t'\rangle$ such that $r'_p(t') = r_p(t)$, $(\mathscr{R},r',t') \models \neg \mathsf{S}^{\boxdot}_{\mathscr{S}}\varphi$. Suppose for a contradiction that $(\mathscr{R},r',t') \models \mathsf{S}^{\boxdot}_{\mathscr{S}}\varphi$. Since $(\mathscr{R},r,t) \models \mathsf{K}_p(p \in \mathscr{S})$, $p \in \mathscr{S}(r',t')$. By positive introspection, $(\mathscr{R},r',t') \models \mathsf{K}_p \mathsf{S}^{\boxdot}_{\mathscr{S}}\varphi$. Thus, $(\mathscr{R},r,t) \models \mathsf{S}^{\boxdot}_{\mathscr{S}}\varphi$, a contradiction. □

The following lemma, while it refers to neither continual common knowledge nor negative introspection, allows Theorem 3 to be applied more widely for weak forms of knowledge:

Lemma 4. *Let $\mathscr{S}$ be an indexical set such that $p \in \mathscr{N} \Rightarrow p \in \mathscr{S}$ is a fact about p's local state (i.e., p can tell whether or not its being nonfaulty would put it in $\mathscr{S}$). Then $p \in \mathscr{S} \wedge \mathsf{B}^{\mathscr{S}}_p\varphi \Rightarrow \mathsf{B}^{\mathscr{N}}_p\varphi$ is valid.*

Proof. Assume that the hypotheses of the lemma hold and suppose that $p \in \mathscr{S} \wedge \mathsf{B}^{\mathscr{S}}_p\varphi$ hold at some point $\langle r,t\rangle$. The fact $p \in \mathscr{N} \Rightarrow p \in \mathscr{S}$ is true at $\langle r,t\rangle$. Consider any point $\langle r',t'\rangle$ such that $r'_p(t') = r_p(t)$ and $p \in \mathscr{N}(r',t')$. Since $p \in \mathscr{N} \Rightarrow p \in \mathscr{S}$ is a fact about p's local state, it is also true at $\langle r',t'\rangle$. This means $p \in \mathscr{S}(r',t')$. Because $\mathsf{B}^{\mathscr{S}}_p\varphi$ held at $\langle r,t\rangle$, φ holds at $\langle r',t'\rangle$. This implies that $\mathsf{B}^{\mathscr{N}}_p\varphi$ holds at $\langle r,t\rangle$. □

Each form of continual common knowledge satisfies an induction rule:

- If $\varphi \Rightarrow \boxdot \mathsf{E}_{\mathcal{G}}(\varphi \wedge \psi)$ is valid in a system, then $\varphi \Rightarrow \mathsf{S}^{\boxdot}_{\mathcal{G}}\psi$ is also valid in that system.
- If $\varphi \Rightarrow \boxdot \mathsf{A}_{\mathcal{G}}(\varphi \wedge \psi)$ is valid in a system, then $\varphi \Rightarrow \mathsf{W}^{\boxdot}_{\mathcal{G}}\psi$ is also valid in that system.

(Again, these follow from the fixed-point definitions.) Finally, continual common knowledge is *continual* in that, if it is true at any point in a run, it is true at every point in that run:

Theorem 5. *The following are valid:*

- $\mathsf{S}^{\boxdot}_{\mathcal{G}}\varphi \Leftrightarrow \boxdot \mathsf{S}^{\boxdot}_{\mathcal{G}}\varphi$;
- $\mathsf{W}^{\boxdot}_{\mathcal{G}}\varphi \Leftrightarrow \boxdot \mathsf{W}^{\boxdot}_{\mathcal{G}}\varphi$.

Proof. Follows directly from the fixed-point axioms and the fact that $\mathsf{K}_p(\varphi \wedge \xi) \Rightarrow \mathsf{K}_p(\varphi)$. □

5. Knowledge and coordination

This section shows some basic relationships between processor knowledge and solutions to the different types of coordination problems defined earlier. These relationships will be used to construct some very simple solutions to these problems that serve as the foundation of subsequent results.

Note first that, to perform an action, a processor must know (or believe) that the action is enabled:

Theorem 6. *Let $\mathscr{C}$ be a coordination problem.*

- *If $P(\Phi)$ P-satisfies $\mathscr{C}$, then $\mathscr{R}_P \models \Phi_{i,p} \Rightarrow \mathsf{B}_p^{\mathcal{N}} ok_i$.*
- *If $P(\Phi)$ C-satisfies $\mathscr{C}$, then $\mathscr{R}_P \models \Phi_{i,p} \Rightarrow \mathsf{K}_p ok_i$.*

Proof. For the first case, suppose for a contradiction that, for some point $\langle r,t\rangle$, $(\mathscr{R}_P, r, t) \models \Phi_{i,p} \wedge \neg \mathsf{B}_p^{\mathcal{N}} ok_i$. Then there must be some point $\langle r',t'\rangle$ such that $r'_p(t') = r_p(t)$ and $(\mathscr{R}_P, r', t') \models p \in \mathcal{N} \wedge \neg ok_i$. Since $\Phi_{i,p}$ is a fact about p's local state, $(\mathscr{R}_P, r', t') \models \Phi_{i,p}$. This contradicts the fact that $P(\Phi)$ P-satisfies $\mathscr{C}$; $\langle r', t\rangle$ is a point by which nonfaulty processor p has performed a_i despite the fact that ok_i is false.

The proof of the second case is similar, except that $\mathsf{K}_p ok_i$ can be shown because ok_i must be true if p performs a_i, even if p is faulty. □

For problems requiring termination, a processor must be sure, before taking an action, that all nonfaulty processors will eventually perform the same action. Each of these processors must in turn know or believe the same thing. This indicates that eventual common knowledge is necessary for problems requiring termination. To prove this, we begin with the following lemma:

Lemma 7. *Let $\mathscr{C}$ be a coordination problem.*

- *If $P(\Phi)$ PT-satisfies $\mathscr{C}$, then $\mathscr{R}_P \models \Phi_{i,p} \wedge (p \in \mathcal{N}) \Rightarrow \mathsf{W}^{\Diamond}_{\mathcal{N}} ok_i$.*
- *If $P(\Phi)$ CT-satisfies $\mathscr{C}$, then $\mathscr{R}_P \models \Phi_{i,p} \Rightarrow \mathsf{S}^{\Diamond}_{\mathcal{N}} ok_i$.*

Proof. For the first case, let $\varphi_i \equiv \bigvee_{p \in \mathcal{N}} \Phi_{i,p}$; φ_i is true at a point if and only if some nonfaulty processor has performed a_i by that point. Since $P(\Phi)$ PT-satisfies $\mathscr{C}$, φ_i implies that all nonfaulty processors eventually perform a_i. Thus, each processor eventually knows that, if it is nonfaulty, $\varphi_i \wedge ok_i$ holds (φ_i by its definition and ok_i by Theorem 6). Thus, $\varphi_i \Rightarrow \Diamond \mathsf{A}_{\mathcal{N}}(\varphi_i \wedge ok_i)$ is valid in $\mathscr{R}_P$. By induction, $\varphi_i \Rightarrow \mathsf{W}^{\Diamond}_{\mathcal{N}} ok_i$ is also valid in $\mathscr{R}_P$. Since $\Phi_{i,p} \wedge (p \in \mathcal{N}) \Rightarrow \varphi_i$, we have $\mathscr{R}_P \models \Phi_{i,p} \wedge (p \in \mathcal{N}) \Rightarrow \mathsf{W}^{\Diamond}_{\mathcal{N}} ok_i$.

The proof of the second case is similar, except that φ_i is defined instead as $\bigvee_{p \in \mathscr{P}} \Phi_{i,p}$. Since $P(\Phi)$ CT-satisfies $\mathscr{C}$, this weaker φ_i also implies that all nonfaulty processors eventually perform a_i. In this case, any nonfaulty processor that executes a_i unconditionally knows φ_i. Furthermore, by Theorem 6, it also knows ok_i. Thus, $\varphi_i \Rightarrow \Diamond \mathsf{E}_{\mathcal{N}}(\varphi_i \wedge ok_i)$ is valid in the system; thus, by induction, $\mathscr{R}_P \models \Phi_{i,p} \Rightarrow \mathsf{S}^{\Diamond}_{\mathcal{N}} ok_i$. □

Arguments similar to those in the proof of Theorem 6 can now be used to show that processors performing an action must actually know (or believe) the required eventual common knowledge:

Theorem 8. *Let $\mathscr{C}$ be a coordination problem.*

- *If $P(\Phi)$ PT-satisfies $\mathscr{C}$, then $\mathscr{R}_P \models \Phi_{i,p} \Rightarrow \mathsf{B}^{\mathcal{N}}_p \mathsf{W}^{\Diamond}_{\mathcal{N}} ok_i$.*
- *If $P(\Phi)$ CT-satisfies $\mathscr{C}$, then $\mathscr{R}_P \models \Phi_{i,p} \Rightarrow \mathsf{K}_p \mathsf{S}^{\Diamond}_{\mathcal{N}} ok_i$.*

Proof. For the first case, suppose for a contradiction that, for some point $\langle r,t \rangle$, $(\mathscr{R}_P, r, t) \models \Phi_{i,p} \wedge \neg \mathsf{B}^{\mathcal{N}}_p \mathsf{W}^{\Diamond}_{\mathcal{N}} ok_i$. Then there must be some point $\langle r',t' \rangle$ such that $r'_p(t') = r_p(t)$ and $(\mathscr{R}_P, r', t') \models p \in \mathcal{N} \wedge \neg \mathsf{W}^{\Diamond}_{\mathcal{N}} ok_i$. Since $\Phi_{i,p}$ is a fact about p's local state, $(\mathscr{R}_P, r', t') \models \Phi_{i,p}$. Point $\langle r',t' \rangle$ is such that $(\mathscr{R}_P, r', t') \models \neg(\Phi_{i,p} \wedge (p \in \mathcal{N}) \Rightarrow \mathsf{W}^{\Diamond}_{\mathcal{N}} ok_i)$, which contradicts Lemma 7. The proof of the second case is similar. □

Theorems 6 and 8 give the conditions necessary for different types of coordination to take place. To simplify the sequel, we introduce the following abbreviations:

$$
\begin{aligned}
&Nec_i(P) \equiv ok_i; && Know_{i,p}(P) \equiv \mathsf{B}^{\mathcal{N}}_p ok_i;\\
&Nec_i(C) \equiv ok_i; && Know_{i,p}(C) \equiv \mathsf{K}_p ok_i;\\
&Nec_i(PT) \equiv \mathsf{W}^{\Diamond}_{\mathcal{N}} ok_i; && Know_{i,p}(PT) \equiv \mathsf{B}^{\mathcal{N}}_p \mathsf{W}^{\Diamond}_{\mathcal{N}} ok_i;\\
&Nec_i(CT) \equiv \mathsf{S}^{\Diamond}_{\mathcal{N}} ok_i. && Know_{i,p}(CT) \equiv \mathsf{K}_p \mathsf{S}^{\Diamond}_{\mathcal{N}} ok_i.
\end{aligned}
$$

Intuitively, $Nec_i(X)$ is a fact that is necessarily true before a processor can perform action a_i in an X-solution to a coordination problem (if X is P or PT, it is only necessary for a correct processor to perform the action). For the P and C cases, this is clear from Validity; for the PT and CT cases, this follows from Lemma 7. $Know_{i,p}(X)$ indicates the knowledge that processor p must have to perform action a_i in an X-solution; this was shown in Theorems 6 and 8. If X is either P or PT, then $Know_{i,p} \equiv \mathsf{B}^{\mathcal{N}}_p(Nec_i(X))$; if X is either C or CT, then $Know_{i,p} \equiv \mathsf{K}_p(Nec_i(X))$.

It is important to note that the necessary conditions to perform an action are stated in terms of processor knowledge and they may or may not hold in a given run. For example, consider a system in which processors always forward all the messages they receive to the other processors and such that no more than $f < n/2$ processors can be faulty by omitting to send or receive messages. In such a system, a processor p receiving $f + 1$ messages each indicating that ok_i holds, would have $Know_{i,p}(CT)$. Intuitively, p knows that one correct processor knows ok_i (one of the $f+1$ messages is from a correct processor), and that all processors will eventually receive $f + 1$ messages indicating ok_i, and so on. If $t > n/2$, then there are runs in which a processor would never be able to have $Know_{i,p}(CT)$.

These necessary conditions suggest a family of very simple coordination protocols. Each endeavors to perform one action as quickly as possible while the others are *never* performed. For each action a_i, we define an action function Φ^i associated with action a_i. The action function is such that processors perform a_i if they have the necessary knowledge to perform a_i. Processors using Φ^i never perform any action other than a_i, which guarantees that processors do not perform conflicting actions. While these protocols may be neither optimal nor optimum, they are important in the development of optimum protocols.

Theorem 9. *Let $\mathscr{C}$ be a coordination problem and let X be either P, C, PT, or CT. For each $a_i \in \mathscr{C}$, consider an action function Φ^i defined as follows:*

$$\Phi^i_{j,p} \equiv \begin{cases} \mathit{false} & \text{if } j \neq i \\ \mathit{Know}_{i,p}(X) & \text{if } j = i \end{cases}$$

for all $p \in \mathscr{P}$. Then, for each i, $P(\Phi^i)$ X-satisfies $\mathscr{C}$.

Proof. Proofs are given for the P and CT cases; the others are similar.

To prove that $P(\Phi^i)$ P-satisfies $\mathscr{C}$, one must show that all runs satisfy the partial Agreement and Validity conditions. Agreement is obviously satisfied: no processor ever performs any action other than a_i. Suppose now that some nonfaulty processor p performs a_i at point $\langle r,t\rangle$. By the definition of $\Phi^i_{i,p}$, $\Phi^i_{i,p} \equiv \mathsf{B}^{\mathscr{N}}_p ok_i$; thus, ok_i must hold at all points $\langle r',t'\rangle$ such that $r'_p(t') = r_p(t)$ and $p \in \mathscr{N}(r')$. $\langle r,t\rangle$ is such a point, so Validity is satisfied.

To prove that $P(\Phi^i)$ CT-satisfies $\mathscr{C}$, one must show that all runs satisfy the complete Agreement, Validity, and Termination conditions. Again, Agreement is trivially satisfied and Validity follows from a proof similar to the above (recall that $\mathsf{S}^{\Diamond}_{\mathscr{N}} ok_i \Rightarrow ok_i$ is valid in the system because $\mathscr{N}$ is never empty). To show Termination, suppose that some processor p performs a_i at point $\langle r,t\rangle$. By the definition of $\Phi^i_{i,p}$, $(\mathscr{R}_P, r, t) \models \mathsf{S}^{\Diamond}_{\mathscr{N}} ok_i$. By the positive introspection of strong eventual common knowledge, it is the case that, for any $q \in \mathscr{N}(r)$, there is some $t' \geqslant t$ such that $(\mathscr{R}_P, r, t') \models \mathsf{K}_q \mathsf{S}^{\Diamond}_{\mathscr{N}} ok_i$. Since this is $\Phi^i_{i,q}$, it should be clear that any processor nonfaulty in run r will eventually perform a_i, and Termination is satisfied. □

6. Optimum protocols

Moses and Tuttle [16] showed that there is no P-optimum protocol for *Eventual Byzantine Agreement*. This section shows there are some coordination problems for which optimum protocols do exist. It precisely characterizes these problems and gives specifications of optimum solutions. The characterization is sufficient because it guarantees that a full-information protocol (see below) can perform an action as soon as any one of the conditions shown necessary by Theorems 6 and 8 becomes true for any full-information protocol (the resulting protocol must thus be optimum). The characterization is necessary because it is implied by the existence of a protocol that dominates all the simple protocols given in Theorem 9 (an optimum protocol would dominate all of these).

As has been noted elsewhere [2, 5, 16, 18], there is an optimum solution to a problem if and only if there is an optimum solution using a *full-information communication protocol* [2, 9, 12]. A full-information protocol is one in which each processor sends its local state to all others each time it is active, and sets its local state to the vector of messages received at the end of its step. Moses and Tuttle [16] showed that, if failures are benign, a full-information protocol can be simulated by one that uses messages of polynomial size. (The cited papers using full-information considered only synchronous systems; nevertheless, their use in the derivation of optimum and optimal solutions is equally valid in asynchronous systems.)

Lemma 10 shows that the conditions shown necessary in Theorems 6 and 8 must be continual common knowledge whenever any action is taken by an *optimum* full-information protocol. More precisely, whenever some action a_i is performed, it is continual common knowledge to the set of processors that might perform another action that conditions necessary for a_i to be performed hold. To get an intuitive feeling for this, consider an optimum protocol P and assume that p is about to perform a_i. It follows that any other processor that performs an action must also perform a_i. Now, consider processor q that has necessary knowledge to perform a_j, $j \neq i$. If q were using Φ^j as defined in Theorem 9, it could perform a_j whenever it has the necessary knowledge to perform a_j. Since P is optimum, it must have q perform some action by this time. Thus, q will perform a_i by that time.

In the statement of this lemma, ψ_i indicates that some (nonfaulty) processor is performing action a_i. (Note that ψ_i is defined to depend only on $\mathcal{N}$ in the simple cases; in the uniform cases, ψ_i depends on all of $\mathcal{P}$.) The sets $\mathcal{S}_i$ to which the continual common knowledge is ascribed contain all processors that have the minimum knowledge necessary to perform some *other* action.

Lemma 10. *Let $\mathscr{C}$ be a coordination problem and let $F(\Psi)$ be a full-information action protocol. Then*

- *If X is either P or PT and $F(\Psi)$ is X-optimum for $\mathscr{C}$, then $\mathscr{R}_F \models \psi_i \Rightarrow \mathsf{W}^{\boxdot}_{\mathcal{S}_i}(Nec_i(X))$, where $\psi_i \equiv \bigvee_{p \in \mathcal{N}} \Psi_{i,p}$ and, for all points $\langle r, t\rangle$ of $\mathscr{R}_F$, $\mathcal{S}_i(r,t) = \{p \in \mathcal{N} \mid \bigvee_{j \neq i} Know_{j,p}(X)\}$.*

- *If X is either C or CT and $F(\Psi)$ is X-optimum for $\mathscr{C}$, then $\mathscr{R}_F \models \psi_i \Rightarrow \mathsf{S}^{\boxdot}_{\mathscr{S}_i}(Nec_i(X))$, where $\psi_i \equiv \bigvee_{p\in\mathscr{P}} \Psi_{i,p}$ and, for all points $\langle r,t\rangle$ of $\mathscr{R}_F$, $\mathscr{S}_i(r,t) = \{p \in \mathscr{P} \mid \bigvee_{j\neq i} Know_{j,p}(X)\}$.*

Proof. The proofs of the four cases are similar; only the PT case is given here. Suppose that $F(\Psi)$ is PT-optimum for $\mathscr{C}$ and that $(\mathscr{R}_F, r, t) \models \psi_i$ for some point $\langle r,t\rangle$. This means that some nonfaulty processor performs a_i by time t in run r. By Agreement, $F(\Psi)$ can have no nonfaulty processor perform any action in r other than a_i. The proof now shows that $(\mathscr{R}_F, r, t) \models \boxdot \mathsf{A}_{\mathscr{S}_i}(\psi_i \wedge \mathsf{W}^{\diamondsuit}_{\mathscr{N}} ok_i)$. Consider any t' and let $p \in \mathscr{S}_i(r,t')$; this means that $p \in \mathscr{N}(r)$ and $(\mathscr{R}_F, r, t') \models Know_{j,p}(PT)$ for some $j \neq i$. Recall the protocol $F(\Phi^j)$ as defined in PT-case of Theorem 9 above (using the full-information communication protocol F). Since $\Phi^j_{j,p} \equiv Know_{j,p}(PT)$, $F(\Phi^j)$ has p act by point $\langle r,t'\rangle$. Since $F(\Psi)$ is PT-optimum and p is nonfaulty, $F(\Psi)$ must also have p act at point $\langle r,t'\rangle$. Because only action a_i can be performed in run r, it must be that $(\mathscr{R}_F, r, t') \models \Psi_{i,p}$. Since $(\Psi_{i,p} \wedge p \in \mathscr{N}) \Rightarrow \psi_i$ and $\mathsf{B}^{\mathscr{N}}_p(p \in \mathscr{N})$ are always valid, $(\mathscr{R}_F, r, t') \models \mathsf{B}^{\mathscr{N}}_p \psi_i$. Furthermore, $(\mathscr{R}_F, r, t') \models \mathsf{B}^{\mathscr{N}}_p(Nec_i(PT))$ by Theorem 8. This means that $(\mathscr{R}_F, r, t') \models \mathsf{B}^{\mathscr{N}}_p(\psi_i \wedge Nec_i(PT))$. Since t' and p were chosen arbitrarily, $(\mathscr{R}_F, r, t) \models \boxdot \mathsf{A}_{\mathscr{S}_i}(\psi_i \wedge Nec_i(PT))$. Since $\langle r,t\rangle$ was chosen arbitrarily, we have $\mathscr{R}_F \models \psi_i \Rightarrow \boxdot \mathsf{A}_{\mathscr{S}_i}(\psi_i \wedge Nec_i(PT))$. By induction, then, $\mathscr{R}_F \models \psi_i \Rightarrow \mathsf{W}^{\boxdot}_{\mathscr{S}_i}(Nec_i(PT))$, as desired. □

Lemma 10 gives a property that holds of any optimum protocol: whenever some action a_i is performed, it is continual common knowledge to the set of processors that might perform another action that conditions necessary for a_i to be performed hold. This property does not hold for all coordination problems. Theorem 11 gives the conditions that are necessary and sufficient for the existence of optimum protocols. Informally, these conditions state that, whenever a processor has the minimum knowledge necessary to perform some action, then it also knows (or believes) that the continual common knowledge given in Lemma 10 holds.

Theorem 11. *Let $\mathscr{C}$ be a coordination problem.*

- *If X is either P or PT, then there is an X-optimum protocol for $\mathscr{C}$ if and only if $\mathscr{R}_F \models Know_{i,p}(X) \Rightarrow \bigvee_{a_j\in\mathscr{C}} \mathsf{B}^{\mathscr{N}}_p \mathsf{W}^{\boxdot}_{\mathscr{S}_j}(Nec_j(X))$, where $\mathscr{S}_j = \{q \in \mathscr{N} \mid \bigvee_{k\neq j} Know_{k,q}(X)\}$.*
- *If X is either C or CT, then there is an X-optimum protocol for $\mathscr{C}$ if and only if $\mathscr{R}_F \models Know_{i,p}(X) \Rightarrow \bigvee_{a_j\in\mathscr{C}} \mathsf{K}_p \mathsf{S}^{\boxdot}_{\mathscr{S}_j}(Nec_j(X))$, where $\mathscr{S}_j = \{q \in \mathscr{P} \mid \bigvee_{k\neq j} Know_{k,q}(X)\}$.*

Proof. We present only the proof of the PT case; the others are similar. The proof proceeds as follows. We first give a protocol $F(\Phi)$, then we show that $F(\Phi)$ both PT-satisfies $\mathscr{C}$ and is PT-optimum. This means that $F(\Phi)$ is a PT-optimum protocol for $\mathscr{C}$. Next, we show that if there is a PT-optimum protocol $F(\Phi)$, then $\mathscr{R}_F \models Know_{i,p}(PT) \Rightarrow \bigvee_{a_j\in\mathscr{C}} \mathsf{B}^{\mathscr{N}}_p \mathsf{W}^{\boxdot}_{\mathscr{S}_j}(Nec_j(PT))$.

Let '$<$' be a total order on the actions in $\mathscr{C}$. Consider first the "if" direction and assume that the given condition holds. We define a protocol $F(\Phi)$ as follows:

$$\Phi_{i,p} \equiv \mathsf{B}_p^{\mathscr{N}}(Nec_i(PT) \wedge \mathsf{W}_{\mathscr{S}_i}^{\boxdot}(Nec_i(PT)) \wedge \bigwedge_{j<i} \neg \mathsf{W}_{\mathscr{S}_j}^{\boxdot}(Nec_j(PT))).$$

We next show that $F(\Phi)$ PT-satisfies $\mathscr{C}$. Consider first Validity. Since $Nec_i(PT) \equiv \mathsf{W}_{\mathscr{N}}^{\Diamond} ok_i$ and $\mathsf{W}_{\mathscr{N}}^{\Diamond} ok_i \Rightarrow ok_i$ (see Section 4.2 above), $\mathscr{R}_F \models \Phi_{i,p} \Rightarrow \mathsf{B}_p^{\mathscr{N}} ok_i$. Thus, $F(\Phi)$ satisfies Validity.

Next consider Agreement. Suppose for a contradiction that, in some run r of $F(\Phi)$, two nonfaulty processors p and q perform actions a_i and a_j at times t and t', respectively, where $i<j$. This implies $(\mathscr{R}_F, r, t') \models \neg \mathsf{W}_{\mathscr{S}_i}^{\boxdot}(Nec_i(PT))$ and $(\mathscr{R}_F, r, t) \models \mathsf{W}_{\mathscr{S}_i}^{\boxdot}(Nec_i(PT))$, which are contradictory because continual common knowledge holds (or fails to hold) at all points of a run.

To prove Termination, suppose that $(\mathscr{R}_F, r, t) \models \Phi_{i,p}$, where $p \in \mathscr{N}(r)$. The proof must show that, for all $q \in \mathscr{N}(r)$, there is some t' such that $(\mathscr{R}_F, r, t') \models \Phi_{i,q}$. By the definition of Φ and the continuality of continual common knowledge, $(\mathscr{R}_F, r, t) \models \boxdot \bigwedge_{j<i} \neg \mathsf{W}_{\mathscr{S}_j}^{\boxdot}(Nec_j(PT))$. Since $Nec_i(PT)$ holds at $\langle r, t\rangle$ and $Nec_i(PT) \equiv \mathsf{W}_{\mathscr{N}}^{\Diamond} ok_i$, $\mathsf{B}_q^{\mathscr{N}}(Nec_i(PT)) \equiv Know_{i,q}(PT)$ must hold at some point $\langle r, t'\rangle$. By definition of $\mathscr{S}_j$, $q \in \mathscr{S}_j(r, t')$ for all $j \neq i$. Note that $q \in \mathscr{N} \Rightarrow q \in \mathscr{S}_j$ is a fact about q's local state. This means that negative introspection for weak continual common knowledge (specifically, the combination of Theorem 3 and Lemma 4) imply $(\mathscr{R}_F, r, t') \models \mathsf{B}_q^{\mathscr{N}}(\bigwedge_{j<i} \neg \mathsf{W}_{\mathscr{S}_j}^{\boxdot}(Nec_j(PT)))$. Since $(\mathscr{R}_F, r, t') \models \mathsf{B}_q^{\mathscr{N}}(Nec_i(PT))$, the hypothesis indicates that $(\mathscr{R}_F, r, t') \models \bigvee_{a_j \in \mathscr{C}} \mathsf{B}_q^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_j}^{\boxdot}(Nec_j(PT))$. If $(\mathscr{R}_F, r, t') \models \mathsf{B}_q^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_j}^{\boxdot}(Nec_j(PT))$ for some $j \neq i$, then, because $q \in \mathscr{S}_j(r, t')$, $(\mathscr{R}_F, r, t') \models \mathsf{B}_q^{\mathscr{N}}(Nec_j(PT))$ holds by positive introspection for continual common knowledge. This means that $q \in \mathscr{S}_i(r, t)$, so $(\mathscr{R}_F, r, t') \models \mathsf{B}_q^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_i}^{\boxdot}(Nec_i(PT))$, again by positive introspection. If $(\mathscr{R}_F, r, t') \models \neg \mathsf{B}_q^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_j}^{\boxdot}(Nec_j(PT))$ for all $j \neq i$, then it must be that $(\mathscr{R}_F, r, t') \models \mathsf{B}_q^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_i}^{\boxdot}(Nec_i(PT))$. In either case, $(\mathscr{R}_F, r, t') \models \mathsf{B}_q^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_i}^{\boxdot}(Nec_i(PT))$. This means that $(\mathscr{R}_F, r, t') \models \Phi_{i,q}$, completing the proof of Termination.

The last requirement of the "if" part of the proof is to show that $F(\Phi)$ is PT-optimum. Consider any full-information protocol $F(\Psi)$ that PT-satisfies $\mathscr{C}$. Suppose that $\Psi_{i,p}$ holds at some point $\langle r, t\rangle$ for some $p \in \mathscr{N}(r)$. The proof must show that, for some $a_j \in \mathscr{C}$, $(\mathscr{R}_F, r, t) \models \Phi_{j,p}$. By Theorem 8 applied to $F(\Psi)$, $(\mathscr{R}_F, r, t) \models Know_{i,p}(PT)$. By the original hypothesis, $(\mathscr{R}_F, r, t) \models \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_j}^{\boxdot}(Nec_j(PT))$ for some $a_j \in \mathscr{C}$. Consider the least such j; we show that $(\mathscr{R}_F, r, t) \models Know_{j,p}(PT)$. If $j = i$, then this is immediate. If $j \neq i$, then $p \in \mathscr{S}_j(r, t)$, so $(\mathscr{R}_F, r, t) \models \mathsf{B}_p^{\mathscr{N}}(Nec_j(PT))$ by positive introspection for continual common knowledge. Now consider any $k<j$; $(\mathscr{R}_F, r, t) \models \neg \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_k}^{\boxdot}(Nec_k(PT))$ by the minimality of j. Since $(\mathscr{R}_F, r, t) \models Know_{j,p}(PT)$ and $p \in \mathscr{N}(r)$, $p \in \mathscr{S}_k(r, t)$. By positive introspection again, it must be that $\neg \mathsf{W}_{\mathscr{S}_k}^{\boxdot}(Nec_j(PT))$. Since $p \in \mathscr{N} \Rightarrow p \in \mathscr{S}_k$ is a fact about p's local state, negative introspection (the combination of Theorem 3 and Lemma 4) implies $\mathsf{B}_p^{\mathscr{N}} \neg \mathsf{W}_{\mathscr{S}_k}^{\boxdot}(Nec_j(PT))$. Since this is true for all $k<j$, it is clear that $\Phi_{j,p}$ holds, as desired.

Finally, consider the "only if" direction. Assume that there is some PT-optimum protocol $F(\Psi)$ for $\mathscr{C}$. The proof must show that $\mathscr{R}_F \models Know_{i,p}(PT) \Rightarrow \bigvee_{a_j \in \mathscr{C}} \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_j}^{\boxdot}(Nec_j(PT))$. Suppose that $(\mathscr{R}_F, r, t) \models Know_{i,p}(PT)$. Consider now two cases:

- For all points $\langle r', t' \rangle$ with $r_p'(t') = r_p(t)$, $p \notin \mathscr{N}(r')$. This means that $(\mathscr{R}_F, r, t) \models \mathsf{B}_p^{\mathscr{N}} \varphi$ for all facts φ, and the result holds trivially.
- For some point $\langle r', t' \rangle$ with $r_p'(t') = r_p(t)$, $p \in \mathscr{N}(r')$. Note that, since p has the same local state at the two points, it believes the same facts at the two points. Thus, $(\mathscr{R}_F, r', t') \models Know_{i,p}(PT)$. This means that, when executing the protocol $F(\Phi^i)$ defined in the PT-case of Theorem 9, p executes a_i by time t' in r'. Since $F(\Psi)$ is PT-optimum, it P-dominates $F(\Phi^i)$, so $(\mathscr{R}_F, r', t') \models \Psi_{j,p}$ for some $a_j \in \mathscr{C}$. Then $(\mathscr{R}_F, r', t') \models \psi_j$, where $\psi_j \equiv \bigvee_{p \in \mathscr{N}} \Psi_{j,p}$. By Lemma 10, $(\mathscr{R}_F, r', t') \models \mathsf{W}_{\mathscr{S}_j}^{\boxdot}(Nec_j(PT))$. Thus, $(\mathscr{R}_F, r, t) \models \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_j}^{\boxdot}(Nec_j(PT))$, which implies $(\mathscr{R}_F, r, t) \models \bigvee_{a_j \in \mathscr{C}} \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_j}^{\boxdot}(Nec_j(PT))$.

Since $\langle r, t \rangle$ was chosen arbitrarily, this gives the desired Validity. □

As an application of Theorem 11, we now show that there can be no P-optimum protocol for *Eventual Byzantine Agreement*. *Eventual Byzantine Agreement* is a coordination problem with two actions a_0 and a_1, where a_i indicates "decide i". The enabling conditions ok_i are "some processor began in initial state i". The problem, as typically defined, considers P-solutions with a stronger notion of termination: the correct processors must perform actions in every run. Since we do not consider problems with strong termination conditions in this paper, we only provide a proof for P- and PT-optimum protocols. Moses and Tuttle proved the same result for the *Eventual Byzantine Agreement* problem with the strong termination condition.

Corollary 12. *There is no P- or PT-optimum protocol for* Eventual Byzantine Agreement.

Proof. We provide the proof for the P case; the PT case is almost identical. Consider a run r in which there are two nonfaulty processors p and q such that p's initial state is 0 and q's is 1. We will show that

$$(\mathscr{R}_F, r, 0) \models Know_{0,p}(P) \wedge \neg \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_0}^{\boxdot}(Nec_0(P)) \wedge \neg \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_1}^{\boxdot}(Nec_1(P)),$$

where $\mathscr{S}_0 = \{\ell \in \mathscr{N} \mid Know_{1,\ell}(P)\}$ and $\mathscr{S}_1 = \{\ell \in \mathscr{N} \mid Know_{0,\ell}(P)\}$. By Theorem 11, this will indicate that there can be no P-optimum protocol for *Eventual Byzantine Agreement*. Consider p's belief at time 0. Clearly, $\mathsf{B}_p^{\mathscr{N}} ok_0$ ($\equiv Know_{0,p}(P)$) holds because p's initial state is 0. This implies $p \in \mathscr{S}_1(r, 0)$. Since there has been no communication, $\neg \mathsf{B}_p^{\mathscr{N}} ok_1$, so it must be that $\neg \mathsf{W}_{\mathscr{S}_1}^{\boxdot} ok_1$ and, therefore, $\neg \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_1}^{\boxdot} ok_1$ or $\neg \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_1}^{\boxdot}(Nec_1(P))$. Similarly, $q \in \mathscr{S}_0(r, 0)$ and $\neg \mathsf{B}_q^{\mathscr{N}} ok_0$, so $\neg \mathsf{W}_{\mathscr{S}_0}^{\boxdot} ok_0$ or $\neg \mathsf{W}_{\mathscr{S}_0}^{\boxdot}(Nec_0(P))$. Since p is nonfaulty in this run, $\neg \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_0}^{\boxdot} ok_0$. Thus, $\neg \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_0}^{\boxdot} ok_0 \wedge \neg \mathsf{B}_p^{\mathscr{N}} \mathsf{W}_{\mathscr{S}_1}^{\boxdot} ok_1$, as desired. □

Theorem 11 gives four conditions, one for each type of coordination, that are necessary and sufficient for the existence of an optimum solution. This theorem can be used to show that certain problems have optimum solutions regardless of the type of coordination required. These include problems whose enabling conditions of all actions are *mutually exclusive*. For example, suppose that one processor is seeking to broadcast a binary value. If a processor can decide on a value v only if that was the broadcaster's value, then the enabling conditions for deciding 0 and deciding 1 are mutually exclusive and there is an optimum protocol.[11] In addition, there may be optimum solutions to problems in which the enabling conditions are related in certain ways. Consider a system and a problem with three possible actions such that $ok_1 \Rightarrow \neg ok_3$ and $ok_2 \Rightarrow ok_1$ are valid in the system. Such a problem always has an optimum solution. To see this, consider Theorem 11 for C solutions and the following cases:

- $(\mathscr{R}_F, r, t) \models \mathit{Know}_{1,p}(C)$. This means $(\mathscr{R}_F, r, t) \models \mathsf{K}_p ok_1$. Consider a point $\langle r', t' \rangle$ with $r'_p(t') = r_p(t)$; then ok_1 holds at $\langle r', t' \rangle$ and ok_3 is false throughout r. Let $j = 1$. $\mathscr{S}_j(r', t') = \{q \in \mathscr{P} \mid (\mathscr{R}_F, r', t') \models \mathsf{K}_q ok_2\}$. Since $ok_2 \Rightarrow ok_1$ (and ok_j is ok_1), it should be clear that $(\mathscr{R}_F, r', t') \models \mathsf{S}^{\boxdot}_{\mathscr{S}_j} ok_j$, as desired.
- $(\mathscr{R}_F, r, t) \models \mathit{Know}_{2,p}(C)$. This means $(\mathscr{R}_F, r, t) \models \mathsf{K}_p ok_2$. Since $ok_2 \Rightarrow ok_1$, $(\mathscr{R}_F, r, t) \models \mathit{Know}_{1,p}(C)$ and the argument above holds (notice that the right sides of the implications in Theorem 11 do not depend on i).
- $(\mathscr{R}_F, r, t) \models \mathit{Know}_{3,p}(C)$. This means $(\mathscr{R}_F, r, t) \models \mathsf{K}_p ok_3$. This means that ok_1 and ok_2 are false throughout r. Letting $j = 3$, we get $\mathscr{S}_j = \emptyset$, and the right side of the implication holds trivially.

Not all coordination problems admit optimum solutions. However, every coordination problem has a nonempty set of *optimal* solutions. The remainder of this paper considers the development of optimal solutions. Halpern et al. [14] showed how the weak form of continual common knowledge could be used to construct P-optimal solutions to *Eventual Byzantine Agreement*. However, they did not explicitly consider the termination properties of the protocols they developed. When problems requiring termination are considered, it is necessary to combine eventual knowledge with continual knowledge. We call this combination *extended knowledge*.

7. Extended knowledge

The optimum protocols given in Section 6 for problems with termination required processors to gain continual common knowledge of eventual common knowledge of some enabling condition. Recall that eventual common knowledge is the greatest fixed point of the "everyone eventually knows" operator, while continual common knowledge is the greatest fixed point of the "everyone always knows" operator. To characterize the domination relation between solutions to problems requiring termination, it becomes

[11] These enabling conditions are different from those classically given for *Reliable Broadcast*. Those conditions also permit deciding either value if the broadcaster is faulty, regardless of its initial value.

essential to develop a form of common knowledge that is the greatest fixed point of both these operators *together*. We call this *extended knowledge*. Extended knowledge pertains to two potentially different sets of processors: the set with eventual knowledge and the set with continual knowledge.

Strong extended knowledge of fact φ with respect to sets $\mathcal{S}$ and $\mathcal{T}$, denoted $\mathsf{S}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi$, is the greatest fixed point of

$$X \Leftrightarrow \Diamond \mathsf{E}_{\mathcal{S}}(\varphi \wedge X) \wedge \boxdot \mathsf{E}_{\mathcal{T}}(\varphi \wedge X).$$

Weak extended knowledge of fact φ with respect to sets $\mathcal{S}$ and $\mathcal{T}$, denoted $\mathsf{W}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi$, is the greatest fixed point of

$$X \Leftrightarrow \Diamond \mathsf{A}_{\mathcal{S}}(\varphi \wedge X) \wedge \boxdot \mathsf{A}_{\mathcal{T}}(\varphi \wedge X).$$

As in Section 4, it is not hard to see that both $\Diamond \mathsf{E}_{\mathcal{S}}(\varphi \wedge X) \wedge \boxdot \mathsf{E}_{\mathcal{T}}(\varphi \wedge X)$ and $\Diamond \mathsf{A}_{\mathcal{S}}(\varphi \wedge X) \wedge \boxdot \mathsf{A}_{\mathcal{T}}(\varphi \wedge X)$ are monotone and thus that both forms of extended common knowledge are well defined. It is also easy to see that $\mathsf{S}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi$ implies the infinite conjunction

$$\Diamond \mathsf{E}_{\mathcal{S}}\varphi \wedge \boxdot \mathsf{E}_{\mathcal{T}}\varphi \wedge (\Diamond \mathsf{E}_{\mathcal{S}})^2\varphi \wedge \Diamond \mathsf{E}_{\mathcal{S}} \boxdot \mathsf{E}_{\mathcal{T}}\varphi \wedge \boxdot \mathsf{E}_{\mathcal{T}} \Diamond \mathsf{E}_{\mathcal{S}}\varphi \wedge (\boxdot \mathsf{E}_{\mathcal{T}})^2\varphi \wedge \cdots. \tag{1}$$

(A similar statement is true of weak extended knowledge.) Extended common knowledge is the first form of common knowledge that is the fixed point of two different knowledge operators. It turns out to be exactly what is necessary to capture the combined Agreement and Termination conditions of some coordination problems.

The two forms of extended knowledge have some important properties that will be useful. Both forms of extended knowledge satisfy positive introspection with respect to both eventual and continual knowledge. That is, it is easy to use the fixed-point definitions to show that the following are valid:

- $\mathsf{S}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi \Rightarrow \Diamond \mathsf{E}_{\mathcal{S}} \mathsf{S}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi$;
- $\mathsf{S}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi \Rightarrow \boxdot \mathsf{E}_{\mathcal{T}}(\varphi \wedge \mathsf{S}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi)$;
- $\mathsf{W}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi \Rightarrow \Diamond \mathsf{A}_{\mathcal{S}} \mathsf{W}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi$; and
- $\mathsf{W}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi \Rightarrow \boxdot \mathsf{A}_{\mathcal{T}}(\varphi \wedge \mathsf{W}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi)$.

(These can be stated more strongly; the forms given are sufficient for the results of this paper.) Both satisfy a kind of negative introspection with respect to continual knowledge, in that the following are valid:

- $\neg \mathsf{S}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi \wedge \mathsf{K}_p(p \in \mathcal{T}) \Rightarrow \mathsf{K}_p \neg \mathsf{S}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi$;
- $\neg \mathsf{W}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi \wedge (p \in \mathcal{T}) \Rightarrow \mathsf{B}^{\mathcal{T}}_p \neg \mathsf{W}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\varphi$.

(The proofs of these are similar to that of Theorem 3.) Extended knowledge does *not* satisfy negative introspection with respect to eventual knowledge, which makes reasoning about it more difficult than reasoning about continual common knowledge.

Each form of extended knowledge satisfies an induction rule that can be used to show that certain facts are extended knowledge:

- If $\varphi \Rightarrow \Diamond \mathsf{E}_{\mathcal{S}}(\varphi \wedge \psi) \wedge \boxdot \mathsf{E}_{\mathcal{T}}(\varphi \wedge \psi)$ is valid in a system, then $\varphi \Rightarrow \mathsf{S}^{\leftrightarrows}_{\mathcal{S},\mathcal{T}}\psi$ is also valid in that system.

- If $\varphi \Rightarrow \Diamond \mathsf{A}_{\mathscr{S}}(\varphi \wedge \psi) \wedge \boxdot \mathsf{A}_{\mathscr{T}}(\varphi \wedge \psi)$ is valid in a system, then $\varphi \Rightarrow \mathsf{W}^{\leftrightarrows}_{\mathscr{S},\mathscr{T}}\psi$ is also valid in that system.

(These follow from the fixed-point definitions.)

The sections below use extended knowledge with the same kinds of sets that Section 6 used with eventual and continual common knowledge. The set used with eventual knowledge is the set $\mathscr{N}$ of nonfaulty processors, because this is the set of processors that must eventually perform an action. The sets used with continual knowledge are sets that know that performing some action is possible; these are the sets that must be brought into agreement with each other. Explicitly considered are cases in which facts φ about runs (specifically, the enabling conditions of a coordination problem) become extended knowledge. Because the first set $\mathscr{N}$ is never empty in the systems we consider, it is not hard to see that, in these systems, $\mathsf{S}^{\leftrightarrows}_{\mathscr{N},\mathscr{S}}\varphi \Rightarrow \varphi$ and $\mathsf{W}^{\leftrightarrows}_{\mathscr{N},\mathscr{S}}\varphi \Rightarrow \varphi$ are valid. These implications will be used to simplify the presentation of some protocols below.

It is important to understand the difference between extended knowledge, as used below, and continual common knowledge of eventual common knowledge, which was used in Section 6. As noted above, strong eventual common knowledge to $\mathscr{S}$ ($\mathsf{S}^{\Diamond}_{\mathscr{S}}\varphi$) implies the infinite conjunction $\bigwedge_{i\geqslant 1}(\Diamond \mathsf{E}_{\mathscr{S}})^i\varphi$. Similarly, strong continual common knowledge to $\mathscr{T}$ ($\mathsf{S}^{\boxdot}_{\mathscr{T}}\varphi$) is equivalent to the infinite conjunction $\bigwedge_{i\geqslant 1}(\boxdot \mathsf{E}_{\mathscr{T}})^i\varphi$. Thus, $\mathsf{S}^{\boxdot}_{\mathscr{T}}\mathsf{S}^{\Diamond}_{\mathscr{S}}\varphi$, which is used in Section 6, implies the infinite conjunction $\bigwedge_{i,j\geqslant 1}(\boxdot \mathsf{E}_{\mathscr{T}})^i(\Diamond \mathsf{E}_{\mathscr{S}})^j\varphi$. Note that all $\boxdot \mathsf{E}_{\mathscr{T}}$ operators precede all $\Diamond \mathsf{E}_{\mathscr{S}}$ operators. In contrast, strong extended knowledge of φ by $\mathscr{S}$ and $\mathscr{T}$ ($\mathsf{S}^{\leftrightarrows}_{\mathscr{S},\mathscr{T}}$) implies the infinite conjunction of all orderings of these operators as noted in Equation 1 above (similar arguments show that $\mathsf{S}^{\leftrightarrows}_{\mathscr{S},\mathscr{T}}$ implies $\mathsf{S}^{\boxdot}_{\mathscr{T}}\mathsf{S}^{\Diamond}_{\mathscr{S}}\varphi$). It is the additional flexibility of the combination of knowledge operators that is needed to characterize domination as is done in the next section.

8. Knowledge and domination

This section exhibits a direct relationship between a dominating protocol and the knowledge that it must have about the protocol it dominates. Both continual common knowledge and extended knowledge are used to characterize this relationship. We then show how these forms of knowledge can be used to construct a protocol that dominates a given protocol.

Theorem 13 shows that the domination relationship between two decision protocols (that use the same communication protocol) can be expressed using some form of common knowledge. In the case of partial solutions, a weak form is used, while a strong form is needed for complete solutions. Solutions with termination use extended knowledge, while the others use only continual common knowledge. Informally, Theorem 13 states that for a processor to perform an action in a *dominating* protocol, it must know (or believe) that the enabling condition for that action is continual common knowledge (or extended knowledge) to some set of processors. The set of processors requiring the knowledge contains exactly those performing some other action in the

dominated protocol. In the case where extended knowledge is needed, the set requiring eventual knowledge is the set of nonfaulty processors.

For each $a_i \in \mathscr{C}$, We define two sets $\mathscr{N}_i$ and $\mathscr{P}_i$. $\mathscr{N}_i$ is the set of nonfaulty processors performing some action other than a_i; $\mathscr{P}_i$ is the set of all processors doing so. Formally, suppose that Φ is the action function used by the protocol to be dominated. $\mathscr{N}_i$ and $\mathscr{P}_i$ are defined as follows. For each system $\mathscr{R}$ and point $\langle r,t\rangle$ of $\mathscr{R}$,

$$\mathscr{N}_i(r,t) = \left\{ p \in \mathscr{N} \mid (\mathscr{R},r,t) \models \bigvee_{j \neq i} \Phi_{j,p} \right\}; \tag{2}$$

$$\mathscr{P}_i(r,t) = \left\{ p \in \mathscr{P} \mid (\mathscr{R},r,t) \models \bigvee_{j \neq i} \Phi_{j,p} \right\}. \tag{3}$$

Because $\Phi_{j,p}$ is a fact about p's local state, it is easy to see that both $(p \in \mathscr{N}_i) \Rightarrow \mathsf{B}_p^{\mathscr{N}}(p \in \mathscr{N}_i)$ and $(p \in \mathscr{P}_i) \Rightarrow \mathsf{K}_p(p \in \mathscr{P}_i)$ are valid.

Having defined the sets $\mathscr{N}_i$ and $\mathscr{P}_i$, we can now give conditions necessary for one protocol to dominate another.

Theorem 13. *Let $\mathscr{C}$ be a coordination problem and let $P(\Phi)$ and $P(\Psi)$ be two decision protocols.*

- *If both protocols P-satisfy $\mathscr{C}$ and $P(\Psi)$ P-dominates $P(\Phi)$, then $\mathscr{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{B}_p^{\mathscr{N}}(ok_i \wedge \mathsf{W}^{\boxdot}_{\mathscr{N}_i} ok_i)$.*
- *If both protocols C-satisfy $\mathscr{C}$ and $P(\Psi)$ C-dominates $P(\Phi)$, then $\mathscr{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{K}_p(ok_i \wedge \mathsf{S}^{\boxdot}_{\mathscr{P}_i} ok_i)$.*
- *If both protocols PT-satisfy $\mathscr{C}$ and $P(\Psi)$ P-dominates $P(\Phi)$, then $\mathscr{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{B}_p^{\mathscr{N}} \mathsf{W}^{\leftrightarrow}_{\mathscr{N},\mathscr{N}_i} ok_i$.*
- *If both protocols CT-satisfy $\mathscr{C}$ and $P(\Psi)$ C-dominates $P(\Phi)$, then $\mathscr{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{K}_p \mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_i} ok_i$.*

($\mathscr{N}_i$ and $\mathscr{P}_i$ are defined with respect to Φ as in Eqs. (2) and (3).)

Proof. Proofs are supplied for the first and fourth cases.

Assume that both protocols P-satisfy $\mathscr{C}$ and that $P(\Psi)$ P-dominates $P(\Phi)$. By Theorem 6, $\mathscr{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{B}_p^{\mathscr{N}} ok_i$. It remains to show that $\mathscr{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{B}_p^{\mathscr{N}} \mathsf{W}^{\boxdot}_{\mathscr{N}_i} ok_i$. Let $\psi_i \equiv \bigvee_{q \in \mathscr{N}} \Psi_{i,q}$. Since $(\Psi_{i,p} \wedge p \in \mathscr{N}) \Rightarrow \psi_i$ and $\mathsf{B}_p^{\mathscr{N}}(p \in \mathscr{N})$ are valid, $\mathscr{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{B}_p^{\mathscr{N}} \psi_i$ (recall that $\Psi_{i,p}$ is a fact about p's local state). Using techniques seen earlier (including induction), it is sufficient to show $\mathscr{R}_P \models \psi_i \Rightarrow \boxdot \mathsf{A}_{\mathscr{N}_i}(\psi_i \wedge ok_i)$. Suppose that $(\mathscr{R}_P,r,t) \models \psi_i$ and assume that $p \in \mathscr{N}_i(r,t')$ for some t'. By the definition of $\mathscr{N}_i$, $p \in \mathscr{N}(r)$ and, for some $j \neq i$, $(\mathscr{R}_P,r,t') \models \Phi_{j,p}$. Since $P(\Psi)$ P-dominates $P(\Phi)$, $P(\Psi)$ must have p perform some action by $\langle r,t'\rangle$; Agreement ensures that it must be a_i, so $(\mathscr{R}_P,r,t') \models \Psi_{i,p}$. By Theorem 6, $(\mathscr{R}_P,r,t') \models \mathsf{B}_p^{\mathscr{N}} ok_i$ and, as argued earlier, $(\mathscr{R}_P,r,t') \models \mathsf{B}_p^{\mathscr{N}} \psi_i$. Thus, $(\mathscr{R}_P,r,t) \models \boxdot \mathsf{A}_{\mathscr{N}_i}(\psi_i \wedge ok_i)$, as desired.

Now assume that both protocols CT-satisfy $\mathscr{C}$ and that $P(\Psi)$ C-dominates $P(\Phi)$. Let $\psi_i \equiv \bigvee_{q \in \mathscr{P}} \Psi_{i,q}$; clearly, $\mathscr{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{K}_p \psi_i$. At this point, it is sufficient to show that $\mathscr{R}_P \models \psi_i \Rightarrow \Diamond \mathsf{E}_{\mathscr{N}}(\psi_i \wedge ok_i) \wedge \boxdot \mathsf{E}_{\mathscr{P}_i}(\psi_i \wedge ok_i)$. Assume that $(\mathscr{R}_P,r,t) \models \psi_i$. The proof

that $(\mathscr{R}_P, r, t) \models \boxdot \mathsf{E}_{\mathscr{P}_i}(\psi_i \wedge ok_i)$ is similar to the one given above for $\mathsf{A}_{\mathscr{N}_i}$. To see that $(\mathscr{R}_P, r, t) \models \Diamond \mathsf{E}_{\mathscr{N}}(\psi_i \wedge ok_i)$, consider some $p \in \mathscr{N}(r)$. Since $P(\Psi)$ has some processor perform a_i in r, p must perform a_i in run r; thus, $(\mathscr{R}_P, r, t') \models \Psi_{i,p}$ for some t'. Clearly, $(\mathscr{R}_F, r, t') \models \mathsf{K}_p \psi_i$. Theorem 6 implies that $(\mathscr{R}_P, r, t') \models \mathsf{K}_p ok_i$, so $(\mathscr{R}_F, r, t') \models \mathsf{K}_p(\psi_i \wedge ok_i)$. Since p was chosen arbitrarily and $\psi_i \wedge ok_i$ is stable, $(\mathscr{R}_F, r, t) \models \Diamond \mathsf{E}_{\mathscr{N}}(\psi_i \wedge ok_i)$, completing the proof. □

(The first case of Theorem 13 generalizes results observed earlier for *Eventual Byzantine Agreement* [14].)

Theorem 13 gives conditions necessary for one protocol to dominate another. To simplify the sequel, we introduce the notations $KnowD_{i,p}(X)$ and $NecD_i(X)$. $KnowD_{i,p}(X)$ is the knowledge (or belief) of processor p if p performs action a_i in protocol $P(\Psi)$ that X-dominate $P(\Phi)$; the facts $NecD_i(X)$ are the facts that must be known (or believed) when p performs action a_i in $P(\Psi)$.

$KnowD_{i,p}(X)$ and $NecD_i(X)$ are defined as follows for the various values of X (P, PT, C, and CT).

$$
\begin{aligned}
NecD_i(\Phi, N) &\equiv \mathsf{W}^{\boxdot}_{\mathscr{N}_i} ok_i; & KnowD_{i,p}(\Phi, N) &\equiv \mathsf{B}^{\mathscr{N}}_p(ok_i \wedge \mathsf{W}^{\boxdot}_{\mathscr{N}_i} ok_i);\\
NecD_i(\Phi, C) &\equiv \mathsf{S}^{\boxdot}_{\mathscr{N}_i} ok_i; & KnowD_{i,p}(\Phi, C) &\equiv \mathsf{K}_p(ok_i \wedge \mathsf{S}^{\boxdot}_{\mathscr{P}_i} ok_i);\\
NecD_i(\Phi, PT) &\equiv \mathsf{W}^{\leftrightarrow}_{\mathscr{N}, \mathscr{N}_i} ok_i; & KnowD_{i,p}(\Phi, PT) &\equiv \mathsf{B}^{\mathscr{N}}_p \mathsf{W}^{\leftrightarrow}_{\mathscr{N}, \mathscr{N}_i} ok_i;\\
NecD_i(\Phi, CT) &\equiv \mathsf{S}^{\leftrightarrow}_{\mathscr{N}, \mathscr{P}_i} ok_i; & KnowD_{i,p}(\Phi, CT) &\equiv \mathsf{K}_p \mathsf{S}^{\leftrightarrow}_{\mathscr{N}, \mathscr{P}_i} ok_i.
\end{aligned}
$$

Theorem 13 shows that, for $P(\Psi)$ to X-dominate $P(\Phi)$, $KnowD_{i,p}(X)$ must hold when $P(\Psi)$ has processor p perform action a_i. All these are facts are the knowledge (or belief) of processor p; the facts $NecD_i(X)$ are the facts that must be known (or believed).

The following lemma is central both to the characterization and construction of optimal protocols given in Section 9. It shows how, given a coordination protocol, to construct another protocol that dominates it. This is done by improving the performance of a selected action so that it is performed as quickly as is possible by any protocol that dominates the original protocol (see Theorem 13).

Lemma 14. *Let $\mathscr{C}$ be a coordination problem and let a_j be any action in $\mathscr{C}$.*

- *If X is either P or PT and $P(\Phi)$ X-satisfies $\mathscr{C}$, then $P(\Psi)$ X-satisfies $\mathscr{C}$ and P-dominates $P(\Phi)$ if*

$$
\Psi_{i,p} \equiv \begin{cases} KnowD_{j,p}(\Phi, X) & \text{if } i = j \\ \Phi_{i,p} \wedge \mathsf{B}^{\mathscr{N}}_p(\neg NecD_j(\Phi, X)) & \text{if } i \neq j. \end{cases}
$$

- *If X is either C or CT and $P(\Phi)$ X-satisfies $\mathscr{C}$, then $P(\Psi)$ X-satisfies $\mathscr{C}$ and C-dominates $P(\Phi)$ if*

$$
\Psi_{i,p} \equiv \begin{cases} KnowD_{j,p}(\Phi, X) & \text{if } i = j \\ \Phi_{i,p} \wedge \mathsf{K}_p(\neg NecD_j(\Phi, X)) & \text{if } i \neq j. \end{cases}
$$

Proof. The proof considers only the CT case; the others are similar. Suppose that $P(\Phi)$ CT-satisfies $\mathscr{C}$ and let Ψ be as defined above. The proof must show that $P(\Psi)$ is a decision protocol that CT-satisfies $\mathscr{C}$ and that it C-dominates $P(\Phi)$.

To prove that $P(\Psi)$ is a decision protocol, we should show that that the action functions $\Psi_{i,p}$ are stable and that processors' choices are unique. To show that $P(\Psi)$ CT-satisfies $\mathscr{C}$, we should show that it satisfies Validity, Agreement, and Termination. The proof that processor choices are unique s is similar to the proof of Agreement (below) and is omitted.

The proof that the action functions are stable is by contradiction. Assume that for some i and p, $\Psi_{i,p}$ is not stable. Consider the following two cases:

- $i=j$. Assume that p performs action j at time t. Then $(\mathscr{R}_P,r,t)\models \mathit{KnowD}_{j,p}(\Phi,CT)$, that is, $(\mathscr{R}_P,r,t)\models \mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$. It is not hard to see that, because ok_j is a fact about the run and the membership of $\mathscr{N}$ is constant throughout a run, $\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$ is stable. This means that $\mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$ is also stable, as desired.
- $i\neq j$. Assume that p performs action i at time t. Then $(\mathscr{R}_P,r,t)\models \Phi_{i,p}\wedge\mathsf{K}_p\neg\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$. Assume that, at time $t_\ell>t$, $(\mathscr{R}_P,r,t_\ell)\models\neg\Psi_{i,p}$. Since $\Phi_{i,p}$ is stable, it follows that $(\mathscr{R}_P,r,t_\ell)\models\neg\mathsf{K}_p\neg\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$. By definition of K_p, there exists a point $\langle r',t'\rangle$ such that $r_p(t_\ell)=r'_p(t')$ and $(\mathscr{R}_P,r',t')\models\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$. Since $(\mathscr{R}_P,r,t_\ell)\models\Phi_{i,p}$ and $\Phi_{i,p}$ is a fact about p's local state, it follows that $(\mathscr{R}_P,r',t')\models\Phi_{i,p}$. Thus, $p\in\mathscr{P}_j(r',t')$. By positive introspection for extended common knowledge, it follows that $(\mathscr{R}_P,r',t')\models\mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$. Since $r_p(t_\ell)=r'_p(t')$, $(\mathscr{R}_P,r,t_\ell)\models\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$. Since $(\mathscr{R}_P,r,t)\models\Phi_{i,p}$, $p\in\mathscr{P}_j(r,t)$ and positive introspection again gives $(\mathscr{R}_P,r,t)\models\mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$. This contradicts $(\mathscr{R}_P,r,t)\models\mathsf{K}_p\neg\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$ from above.

Now, we prove that $P(\Psi)$ CT-satisfies $\mathscr{C}$. Consider first the Validity condition. Suppose that $(\mathscr{R}_P,r,t)\models\Psi_{i,p}$. If $i=j$, then $(\mathscr{R}_P,r,t)\models\mathit{KnowD}_{j,p}(\Phi,CT)$, that is, $(\mathscr{R}_P,r,t)\models\mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$. Since $\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j\Rightarrow ok_j$ is valid in the system, a_j is an enabled action in r. If $i\neq j$, then $(\mathscr{R}_P,r,t)\models\Phi_{i,p}$ and a_i is enabled because of the assumption that $P(\Phi)$ CT-satisfies $\mathscr{C}$ (the correctness of $P(\Phi)$ ensures Validity).

Next, consider Agreement. Suppose that processors p and q perform actions a_i and a_k, respectively, in run r. If neither i nor k is j, then $\Phi_{i,p}$ and $\Phi_{k,q}$ hold in r, implying that $i=k$ (because $P(\Phi)$ correctly gives Agreement). Alternatively, assume without loss of generality that p performs a_j at time t. Then $(\mathscr{R}_P,r,t)\models\mathit{KnowD}_{j,p}(\Phi,CT)$, that is, $(\mathscr{R}_P,r,t)\models\mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$. Suppose for a contradiction that $(\mathscr{R}_P,r,t')\models\Psi_{k,q}$ for some t' and $k\neq j$; by the definition of Ψ, $(\mathscr{R}_P,r,t')\models\Phi_{k,q}$ as well. This implies that $q\in\mathscr{P}_j(r,t')$, so $(\mathscr{R}_P,r,t')\models\mathsf{K}_q\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$ (or $(\mathscr{R}_P,r,t')\models\mathit{KnowD}_{j,q}(\Phi,CT)$) by positive introspection. Thus, $(\mathscr{R}_P,r,t')\models\neg\Psi_{k,q}$, giving the desired contradiction.

Finally, consider Termination. Assume that $(\mathscr{R}_P,r,t)\models\Psi_{i,p}$. The proof must show that, for every $q\in\mathscr{N}(r)$, there is some t' such that $(\mathscr{R}_P,r,t')\models\Psi_{i,q}$. Consider two cases:

- $i=j$. Then $(\mathscr{R}_P,r,t)\models\mathit{KnowD}_{j,p}(\Phi,CT)$, that is, $(\mathscr{R}_P,r,t)\models\mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$. By positive introspection, there is a $t'\geqslant t$ such that $(\mathscr{R}_P,r,t')\models\mathsf{K}_q\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j}ok_j$; i.e. $(\mathscr{R}_P,r,t')\models\mathit{KnowD}_{j,q}(\Phi,CT)$. Since $\Psi_{j,q}\equiv\mathit{KnowD}_{j,q}(\Phi,CT)$, this completes the proof.

- $i \neq j$. Then $(\mathscr{R}_P, r, t) \models \Phi_{i,p} \wedge \mathsf{K}_p \neg \mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j} ok_j$, so $(\mathscr{R}_P, r, t) \models \mathsf{K}_p(\neg NecD_j(\Phi, CT))$. Note that $p \in \mathscr{P}_j(r,t)$. By the Termination of $P(\Phi)$, there is a t' such that $(\mathscr{R}_P, r, t') \models \Phi_{i,q}$. If $(\mathscr{R}_P, r, t') \models \mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j} ok_j$, then $(\mathscr{R}_P, r, t) \models \mathsf{K}_p \mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j} ok_j$ by positive introspection. This is a contradiction, so $(\mathscr{R}_P, r, t') \models \neg \mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j} ok_j$. Since $(\mathscr{R}_P, r, t') \models \mathsf{K}_q(q \in \mathscr{P}_j)$, $(\mathscr{R}_P, r, t') \models \mathsf{K}_q(\neg \mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j} ok_j)$ (i.e. $(\mathscr{R}_P, r, t') \models \mathsf{K}_q(\neg NecD_j(\Phi, CT))$) by negative introspection. Thus, $(\mathscr{R}_P, r, t') \models \Phi_{i,q} \wedge \mathsf{K}_q(\neg NecD_j(\Phi, CT))$. Since $\Psi_{i,q}$ is defined to be $\Phi_{i,q} \wedge \mathsf{K}_q(\neg NecD_j(\Phi, CT))$, this completes the proof.

In either case, q eventually performs a_i, as desired.

To show that $P(\Psi)$ C-dominates $P(\Phi)$, assume that $(\mathscr{R}_P, r, t) \models \Phi_{i,p}$. The proof must show that, for some $a_k \in \mathscr{C}$, $(\mathscr{R}_P, r, t) \models \Psi_{k,p}$. Consider two cases:

- $i = j$. Since the protocol $P(\Phi)$ C-dominates itself, Theorem 13 implies that $(\mathscr{R}_P, r, t) \models KnowD_{j,p}(\Phi, CT)$; thus, $(\mathscr{R}_P, r, t) \models \Psi_{j,p}$.
- $i \neq j$. This implies that $(\mathscr{R}_P, r, t) \models \mathsf{K}_p(p \in \mathscr{P}_j)$. Consider now two sub-cases:
  - $(\mathscr{R}_P, r, t) \models \mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j} ok_j$. Then $(\mathscr{R}_P, r, t) \models \mathsf{K}_p \mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j} ok_j$ by positive introspection. Since $\mathsf{K}_p \mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j} ok_j \equiv KnowD_{j,p}(\Phi, CT) \equiv \Psi_{j,p}$, $(\mathscr{R}_P, r, t) \models \Psi_{j,p}$.
  - $(\mathscr{R}_P, r, t) \models \neg \mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j} ok_j$. Then $(\mathscr{R}_P, r, t) \models \mathsf{K}_p \neg \mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j} ok_j$ by negative introspection. Since $\mathsf{S}^{\leftrightarrow}_{\mathscr{N},\mathscr{P}_j} ok_j \equiv NecD_j(\Phi, CT)$ and $\Psi_{i,p} \equiv \Phi_{i,p} \wedge \mathsf{K}_p(\neg NecD_j(\Phi, CT))$, $(\mathscr{R}_P, r, t) \models \Psi_{i,p}$.

In all cases, $P(\Phi)$ has p act at time t, as desired, so $P(\Psi)$ C-dominates $P(\Phi)$. □

Lemma 14 gives a method by which a protocol $F(\Psi)$ that X-dominates $F(\Phi)$ may be constructed. Denote Ψ by $Dom_j(\Phi, X)$, where a_j is the action whose performance is being optimized. This notation will be used in Section 9 in the construction of optimal protocols.

Halpern et al. [14] gave a different method, using continual common knowledge, to take a P-solution to *Eventual Byzantine Agreement* and construct from it a P-dominating P-solution; their technique can easily be extended to apply to coordination problems that do not require termination. It cannot, however, be applied to problems requiring termination (even by using extended knowledge in place of continual common knowledge). Informally, this is because their techniques rely on negative introspection properties of continual common knowledge that are not possessed by extended knowledge. Thus, the techniques presented here are more general because they can be applied to problems requiring termination. In addition, they are simpler in the following sense: the dominating protocol is constructed by replacing only one action predicate (the one for the action whose performance is to be improved); the others are simply augmented by adding a conjunct to the already existing predicate. This allows the new protocol to rely more directly on the correctness of the original one.

It is important to note that Lemma 14 provides a general way to construct a dominating protocol of a given protocol. The dominating protocol's decision functions are expressed in terms of the knowledge that processors have. In general, determining whether a processor has the required knowledge to act in a dominating protocol is not easy. Methods for doing so would depend on the particular system and coordination problem. Providing a general way to determine if the required knowledge holds is

beyond the scope of this paper. In another work [1], we considered the problem of determining whether eventual common knowledge holds. Developing similar general techniques for other types of knowledge is a subject for future research.

Theorem 13 can be used to show that the generated protocol $P(\Psi)$ performs the chosen action a_j as quickly as *any* protocol that dominates the original $P(\Phi)$. Theorem 16 in the following section shows how Lemma 14 can be used iteratively to generate optimal protocols.

9. Optimal protocols

This section provides a precise characterization of optimal protocols for coordination and a method by which any protocol can be converted into an optimal one. As in Section 6, this section concentrates on full-information protocols, in which a processor sends its local state (as a message) in each round and then sets its local state to the vector of messages it receives. If there is an optimal protocol that dominates a given protocol, then there is an optimal full-information protocol that does so also.

Theorem 15 gives the necessary and sufficient conditions for a full-information protocol to be optimal. These conditions are closely related to the conditions established in Theorem 13.

Theorem 15. *Let $\mathscr{C}$ be a coordination problem, let X be one of P, C, PT, and CT, and let $F(\Phi)$ be a full-information decision protocol that X-satisfies $\mathscr{C}$. $F(\Phi)$ is X-optimal for $\mathscr{C}$ if and only if $\mathscr{R}_F \models \Phi_{i,p} \Leftrightarrow KnowD_{i,p}(\Phi, X) \wedge \bigwedge_{j \neq i} \neg \Phi_{j,p}$.*

Proof. We prove only the "if" direction for P-optimality and the "only if" direction for CT-optimality. The other cases are similar.

Consider first the "if" direction for P-optimality. Assume that

$$\mathscr{R}_F \models \Phi_{i,p} \Leftrightarrow KnowD_{i,p}(\Phi, P) \wedge \bigwedge_{j \neq i} \neg \Phi_{j,p}.$$

Let $F(\Psi)$ be a P-solution to $\mathscr{C}$ that P-dominates $F(\Phi)$; the proof must show that $F(\Phi)$ P-dominates $F(\Psi)$. Suppose that $(\mathscr{R}_F, r, t) \models \Psi_{i,p}$. By Theorem 6, $(\mathscr{R}_F, r, t) \models \mathsf{B}_p^{\mathcal{N}} ok_i$. If $(\mathscr{R}_F, r, t) \models \bigvee_{j \neq i} \Phi_{j,p}$, the proof is complete; assume instead that $(\mathscr{R}_F, r, t) \models \bigwedge_{j \neq i} \neg \Phi_{j,p}$. Since $F(\Psi)$ P-dominates $F(\Phi)$, Theorem 13 implies that $(\mathscr{R}_F, r, t) \models \mathsf{B}_p^{\mathcal{N}} \mathsf{W}_{\mathcal{N}_i}^{\boxdot} ok_i$. Since $KnowD_{i,p}(\Phi, P) \equiv \mathsf{B}_p^{\mathcal{N}}(ok_i \wedge \mathsf{W}_{\mathcal{N}_i}^{\boxdot} ok_i)$, $(\mathscr{R}_F, r, t) \models \Phi_{i,p}$. This implies that $F(\Phi)$ P-dominates $F(\Psi)$ and is thus P-optimal.

Now consider the "only if" direction for CT-optimality. Assume that $F(\Phi)$ is CT-optimal for $\mathscr{C}$; the proof must show that $\mathscr{R}_F \models \Phi_{i,p} \Leftrightarrow KnowD_{i,p}(\Phi, CT) \wedge \bigwedge_{j \neq i} \neg \Phi_{j,p}$. Suppose first that $(\mathscr{R}_F, r, t) \models \Phi_{i,p}$. Since $F(\Phi)$ is a decision protocol, $(\mathscr{R}_F, r, t) \models \bigwedge_{j \neq i} \neg \Phi_{j,p}$. Because $F(\Phi)$ C-dominates itself, Theorem 13 implies that $(\mathscr{R}_F, r, t) \models KnowD_{i,p}(\Phi, P)$, giving the desired implication. Finally, suppose that $(\mathscr{R}_F, r, t) \models KnowD_{i,p}(\Phi, P) \wedge \bigwedge_{j \neq i} \neg \Phi_{j,p}$. Let $F(\Psi)$ be such that Ψ is $Dom_i(\Phi, CT)$ (see the paragraph following the proof of Lemma 14); that is, $F(\Psi)$ CT-satisfies $\mathscr{C}$, C-dominates

$F(\Phi)$, and optimizes for action a_i. Since $\Psi_{i,p}$ is defined to be $KnowD_{i,p}(\Phi, P)$, $(\mathscr{R}_F, r, t \models \Psi_{i,p}$. Since $F(\Phi)$ is CT-optimal, it C-dominates $F(\Psi)$, so $(\mathscr{R}_F, r, t) \models \Phi_{j,p}$ for some $a_j \in \mathscr{C}$. Since all other actions are already excluded, it must be that $(\mathscr{R}_F, r, t) \models \Phi_{i,p}$, completing the proof. □

The P case of Theorem 15 is similar to a result observed earlier for *Eventual Byzantine Agreement* [14].

The characterization of optimal protocols given in Theorem 15, although precise, does not indicate how one might construct such a protocol. This is in marked contrast to work on simultaneous coordination [5, 16, 18]. That work first exhibited the knowledge needed to achieve such coordination and then used it directly to construct optimum solutions. To develop optimal protocols for nonsimultaneous coordination, one can iteratively apply Lemma 14 to some initial protocol. The idea is that each application of the lemma improves the performance of a particular action. After all actions have been improved, the result is an optimal protocol.

Theorem 16. *Let $\mathscr{C}$ be a coordination problem, let '$<$' be some total order of the actions in $\mathscr{C}$ ($a_1 < a_2 < \cdots < a_m$), and let X be either P, C, PT, or CT. Let $F(\Phi)$ be a full-information decision protocol that X-satisfies $\mathscr{C}$. Inductively define Φ^i ($0 \leqslant i \leqslant m$) as follows: Φ^0 is Φ and Φ^{i+1} is $Dom_{i+1}(\Phi^i, X)$ (see the paragraph following the proof of Lemma 14). Then $F(\Phi^m)$ is X-optimal for $\mathscr{C}$ and Y-dominates $F(\Phi)$ (where Y is P if X is P or PT and C if X is C or CT).*

Proof. The proof considers the case of CT-optimality; the others are similar. It follows by Lemma 14 that, for all i ($1 \leqslant i \leqslant m$), $F(\Phi^i)$ CT-satisfies $\mathscr{C}$ and C-dominates all $F(\Phi^j)$ with $j \leqslant i$. It remains only to show that $F(\Phi^m)$ is CT-optimal; this is done by showing that it C-dominates any protocol $F(\Psi)$ that CT-satisfies $\mathscr{C}$ and C-dominates $F(\Phi^m)$. To show that $F(\Phi^m)$ C-dominates $F(\Psi)$, assume that $(\mathscr{R}_F, r, t) \models \Psi_{i,p}$. Since $F(\Psi)$ C-dominates $F(\Phi^m)$, it also C-dominates $F(\Phi^{i-1})$. Thus, by Theorem 13, $(\mathscr{R}_F, r, t) \models KnowD_{i,p}(\Phi^{i-1}, CT)$. But this is precisely $\Phi^i_{i,p}$, so $F(\Phi^i)$ has p perform a_i at $\langle r, t\rangle$. Since $F(\Phi^m)$ C-dominates $F(\Phi^i)$, it also has p perform some action at $\langle r, t\rangle$. Thus, $F(\Phi^m)$ C-dominates $F(\Psi)$ and is CT-optimal. □

Theorem 16 shows how any protocol can be converted into an optimal protocol. In particular, it can be applied to a degenerate protocol $F(\Phi)$ that performs no actions (i.e., with $\Phi_{i,p} = false$ for all i and p). The first application of Lemma 14 results then in the following action function (for CT-satisfaction):

$$\Phi^1_{i,p} \equiv \begin{cases} \mathsf{K}_p \mathsf{S}^{\leftrightarrows}_{\mathscr{N},\mathscr{P}_1} ok_1 & \text{if } i = 1, \\ false & \text{if } i > 1, \end{cases}$$

where $\mathscr{P}_1$ is based on Φ and is thus empty. Thus, $\Phi^1_{1,p}$ simplifies to $\mathsf{K}_p \mathsf{S}^{\Diamond}_{\mathscr{N}} ok_1$. Thus, by Theorem 8, $F(\Phi^1)$ performs a_1 as early as any protocol can (it is one of the protocols

given by Theorem 9). The second application results in the following:

$$\Phi^2_{i,p} \equiv \begin{cases} \mathsf{K}_p(\mathsf{S}^{\Diamond}_{\mathcal{N}} ok_1 \wedge \neg \mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathscr{P}_2} ok_2) & \text{if } i=1, \\ \mathsf{K}_p \mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathscr{P}_2} ok_2 & \text{if } i=2, \\ \mathit{false} & \text{if } i>2, \end{cases}$$

where $\mathscr{P}_2$ is based on Φ^1 and is thus $\{q \in \mathscr{P} \mid \mathsf{K}_q \mathsf{S}^{\Diamond}_{\mathcal{N}} ok_1\}$. If there are only two actions in $\mathscr{C}$ (as in the case of *Eventual Byzantine Agreement*), then $F(\Phi^2)$ is CT-optimal. Otherwise, Lemma 14 can be applied as many times as necessary.

The actual "compilation" of a knowledge-based protocol generated by Theorem 16 into a traditional one that does not explicitly use knowledge would require an analysis based on the semantics of eventual common knowledge, continual common knowledge, or extended knowledge. This is beyond the scope of this paper. Other papers that do study these semantics are discussed in the next section.

10. Discussion and conclusions

This paper considered four different types of coordination problems. For each problem, we determined the knowledge necessary to perform an action and used this to characterize the domination relationship between different solutions and to develop and characterize optimum and optimal solutions. In the past, researchers have used simple common knowledge to study *simultaneous* coordination [5, 16, 18]. When the simultaneity restriction is relaxed, weaker (but less intuitive) variants of common knowledge become more appropriate. These variants are the fixed points of certain knowledge operators. The operators used depended on the type of coordination desired:

- uniform coordination requires knowledge, whereas simple coordination requires only belief;
- the Agreement condition of coordination requires continual knowledge to ensure that there is never a disagreement; and
- the Termination condition of coordination requires eventual knowledge to ensure that all nonfaulty processors eventually decide.

A major contribution of this paper is the definition of *extended knowledge*, which combines the continual and eventual knowledge needed for coordination problems with termination.

Necessary and sufficient conditions were given for the existence of an optimum solution to a problem in a given system; furthermore, a knowledge-based specification of such solutions was given for cases in which the conditions were met. These conditions depended on the type of coordination desired. While some problems have optimum solutions regardless of the type of coordination required, it seems likely that the type of coordination will be important in some cases. Furthermore, it is quite possible that, for some problems, the existence of an optimum solution may depend also on the type and number of failures that can occur or on the synchrony of the communication network.

In the future, we plan to further study these conditions to provide, when possible, a simpler characterization of coordination problems with optimum solutions.

Some of the optimum solutions given require action when some fact becomes eventual common knowledge. A better understanding of the semantics of this knowledge would facilitate the implementation of such protocols and an understanding of their complexity. (Moses and Tuttle [16] analyze the complexity of computing simple common knowledge.) Tuttle [20] gives a characterization of the semantics of eventual common knowledge based on game theory. In a separate paper [1], we study the relationship between eventual common knowledge and *distributed knowledge* [7, 13]. For the purposes of this paper, distributed knowledge of a fact *about the input* turns out to be equivalent to weak eventual common knowledge of the same fact. Because it is easier to reason about distributed knowledge than eventual common knowledge, we can use this equivalence to simplify the implementation and analysis of some of the protocols discussed here. For example, we show that, in systems with general omission failures, testing for distributed knowledge is NP-hard.

There are also cases in which the necessary knowledge is impossible to attain. Neiger and Tuttle [18] showed that strong common knowledge cannot be achieved in synchronous distributed systems with general omission failures in which n, the number of processors, is less than or equal to $2t$, where t is the maximum number of faulty processors. This shows that uniform simultaneous coordination cannot be achieved in these systems. We show that strong eventual common knowledge (of facts about the input) cannot be attained in these same systems, indicating that uniform nonsimultaneous coordination cannot be achieved in these systems. We also show that strong eventual common knowledge cannot be achieved in *asynchronous* systems with send-omission failures in which $n \leqslant 2t$, indicating again that uniform coordination cannot be achieved in such cases. In the future, we plan to study the systems in which the various forms of eventual common knowledge can be achieved so as to better understand when different forms of coordination are possible.

Our development of optimal protocols uses extended knowledge. Implementation of these protocols will depend on gaining a better understanding of this new form of knowledge. It is possible that the semantics of extended knowledge can be understood by combining the game-theoretic characterization of eventual common knowledge [20] with the graph-theoretic characterization of continual common knowledge [14]. Just as Moses and Tuttle [16] showed how a graph-theoretic characterization of common knowledge could be used to implement and analyze the complexity of simultaneous coordination protocols, a better understanding of extended knowledge might be applied to the more general form of coordination considered here.

It should be noted that the results in this paper apply to systems with both synchronous and asynchronous communication. In the past, most papers involving knowledge and coordination have concentrated on systems with synchronous communication. Because we consider a new form of termination that is weaker, but more appropriate to asynchronous systems, than the one used earlier, our analysis applies to these more practical systems. For example, the results of this paper can be applied

to the protocols developed by Gopal and Toueg [11] for coordination in asynchronous systems.

It is reasonable to ask why Halpern et al. [14], who considered a problem that does require termination (*Eventual Byzantine Agreement*), used continual common knowledge, instead of extended common knowledge, to construct optimal protocols. They considered *Eventual Byzantine Agreement* to require *unconditional termination*, meaning that all correct processors terminate in every run. They developed "dominating operators" similar to those presented in our Lemma 14 but based on continual common knowledge instead of extended common knowledge. It should be clear that any protocol that dominates one that terminates unconditionally will also terminate unconditionally. For that reason, their dominating operators were not specially devised to consider termination.

In contrast, this paper considers problems requiring *conditional termination*: a correct processor is required to terminate only if some other processor does so.[11] A protocol that dominates a conditionally terminating protocol might not itself conditionally terminate. For example, suppose that $P(\Phi)$ is conditionally terminating and that, in some operating environment, no processor terminates. If $P(\Psi)$ dominates $P(\Phi)$, it may be, in that same operating environment, that correct processor p terminates but correct processor q does not. This means that $P(\Psi)$ is not conditionally terminating. Because a protocol that dominates one that conditionally terminates might not do so itself, it was necessary for our dominating operators to be constructed so as to ensure the conditional termination of the resulting dominating protocols. A methodology using extended knowledge was needed in order to do this.

Acknowledgements

The authors thank Joseph Y. Halpern, Yoram Moses, Sam Toueg, Mark R. Tuttle, and Orli Waarts for discussions and comments that helped in the development of this work. In addition, we thank the anonymous referees for their careful reading of the paper and extensive comments.

References

[1] R.A. Bazzi, G. Neiger, The complexity and impossibility of achieving fault-tolerant coordination, in: Proc. 11th ACM Symp. on Principles of Distributed Computing, ACM, New York, 1992, pp. 203–214.

[2] B.A. Coan, A communication-efficient canonical form for fault-tolerant distributed protocols, in: Proc. 5th ACM Symp. on Principles of Distributed Computing, August 1986, pp. 63–72. A revised version appears in Coan's Ph.D. dissertation [3].

[3] B.A. Coan, Achieving consensus in fault-tolerant distributed computer systems: protocols, lower bounds, and simulations, Ph.D. dissertation, Massachusetts Institute of Technology, June 1987.

[4] D. Dolev, R. Reischuk, H. Raymond Strong, Eventual is earlier than immediate, in: Proc. 23rd Symp. on Foundations of Computer Science, IEEE Computer Society Press, Silver Spring, 1982, pp. 196–203.

[11] Reasons for considering this form of termination are outlined in Section 3.1.

[5] C. Dwork, Y. Moses, Knowledge and common knowledge in a Byzantine environment: crash failures, Inform. and Comput. 88 (2) (1990) 156–186.
[6] R. Fagin, J.Y. Halpern, Y. Moses, M.Y. Vardi, Reasoning about Knowledge, MIT Press, Cambridge, MA, 1995.
[7] R. Fagin, M.Y. Vardi, Knowledge and implicit knowledge in a distributed environment: preliminary report, in: J.Y. Halpern (Ed.), Proc. 1st Conf. on Theoretical Aspects of Reasoning about Knowledge, Morgan-Kaufmann, LosAltos, CA, 1986, pp. 187–206. Also appears as Technical Report RJ4990, IBM Research Laboratory.
[8] M.J. Fischer, The consensus problem in unreliable distributed systems (a brief survey), in: M. Karpinsky (Ed.), Foundations of Computation Theory, Lecture Notes on Computer Science, vol. 158, Springer, Berlin, 1983, pp.127–140.
[9] M.J. Fischer, N.A. Lynch, A lower bound for the time to assure interactive consistency, Inform. Process. Lett. 14, 1982, pp. 183–186.
[10] M.J. Fischer, N.A. Lynch, M.S. Paterson, Impossibility of distributed consensus with one faulty process, J. ACM 32 (2) (1985) pp. 374–382.
[11] A. Gopal, S. Toueg, Reliable broadcast in synchronous and asynchronous environments, in: J.-C. Bermond, M. Raynal (Eds.), Proc. 3rd Internat. Workshop on Distributed Algorithms, Lecture Notes on Computer Science, vol. 392, Springer, Berlin, 1989, pp. 110–123.
[12] V. Hadzilacos, Issues of Fault Tolerance in Concurrent Computations, Ph.D. dissertation, Harvard University, 1984.
[13] J.Y. Halpern, Y. Moses, Knowledge and common knowledge in a distributed environment, J. ACM 37 (3) (1990) pp. 549–587.
[14] J.Y. Halpern, Y. Moses, O. Waarts, A characterization of eventual Byzantine agreement, in: Proc. 9th ACM Symp. on Principles of Distributed Computing, ACM, New York, 1990, pp. 333–346.
[15] R. Michel, Knowledge in distributed byzantine environments, Ph.D. dissertation, Yale University, New Hoven, 1989.
[16] Y. Moses, M.R. Tuttle, Programming simultaneous actions using common knowledge, Algorithmica 3 (1) (1988) 121–169.
[17] G. Neiger, S. Toueg, Automatically increasing the fault-tolerance of distributed algorithms, J. Algorithms 11 (3) (1990) 374–419.
[18] G. Neiger, M.R. Tuttle, Common knowledge and consistent simultaneous coordination, Distrib. Comput. 6 (3) (1993) 181–192.
[19] M. Pease, R. Shostak, L. Lamport, Reaching agreement in the presence of faults, J. ACM 27 (2) (1980) 228–234.
[20] M.R. Tuttle, A game-theoretic characterization of eventual common knowledge, Unpublished manuscript, October 1988.